MASTER EN SCIENCES INFORMATIQUES

Extraction de données de sites web : méthodologie, outils et étude de cas

Meurisse, Jean-Roch

Award date:
2004

Facultés Universitaires
Notre-Dame de la Paix, Namur
Institut d'Informatique.

Année académique 2003-2004

Extraction de données de sites web : Méthodologie, outils et étude de cas

Jean-Roch Meurisse

Mémoire présenté en vue de l'obtention du grade de licencié en informatique.

Résumé

Dans ce document, nous présentons une méthodologie visant à extraire de sites internet les données et leur structure sémantique. Les pages composant le site sont classées en fonction de leur contenu informationnel en types de pages. Chaque type de pages est décrit dans un document *XML* répondant à un formalisme appelé *Meta*. Ce document répertorie, nomme et hiérarchise les concepts identifiés dans le type de pages et situe les données dans l'arborescence *HTML*. Ce même document est utilisé pour extraire, d'une part, la structure des données sous la forme d'un *XML Schema* et, d'autre part, les données dans un fichier *XML* conforme à cette structure. Les différents *XML Schemas* sont ensuite intégrés dans un schéma conceptuel unique représentant la totalité du domaine d'application. Ce schéma conceptuel sert de point de départ pour la conception d'une base de données qui accueillera les données extraites des pages *HTML*.

La méthodologie est illustrée dans une étude de cas réalisée avec l'aide d'outils existants ou développés spécifiquement pour la démarche.

Mots-clés : extraction de données, extraction de schémas, *XML*, rétro-ingénierie.

Abstract

This document presents a method to extract data and their semantic structures from web sites. The site pages are classified into page types according to their informational content. Each page type is described in an *XML* document produced in accordance with a formalism called *Meta*. In the *Meta* document, on the one hand, each concept identified in a page type is listed, interpreted and organised into a hierarchy, and on the other hand, data are localised in the *HTML* tree. The *Meta* document is then used to extract an *XML Schema* describing the data structure and an *XML Schema*-valid *XML* document containing data gathered from *HTML* pages. All *XML Schemas* are then integrated into a unique conceptual schema that represents the whole application domain. From this conceptual schema, a database is designed in order to record extracted data.

The method is illustrated in a case study using existing and specifically developped tools.

Keywords : data extraction, schema extraction, *XML*, reverse engineering.

Je tiens tout d'abord à remercier Jean-Luc Hainaut, promoteur de ce mémoire, pour l'intérêt qu'il a porté à ce travail, pour ses conseils avisés et pour m'avoir accueilli au sein du Laboratoire d'Ingénierie des Bases de Données.

Ma reconnaissance s'adresse également à Jean Henrard et Fabrice Estiévenart pour leur aide précieuse et leur aimable collaboration lors de la rédaction de ce mémoire.

Je remercie également tous les membres du LIBD pour le bon accueil qu'ils m'ont réservé à mon arrivée.

Merci aussi à mon frère Hubert pour m'avoir poussé à tenter l'aventure de la licence à horaire décalé et pour avoir relu avec attention ce travail.

Enfin, à toute ma famille et mes amis, merci pour le soutien continu que vous m'avez témoigné au cours de ces quatre années d'études.

Table des matières

Chapitre 1

Introduction

1.1 La problématique

L'internet a pris une place importante dans notre quotidien. Il nous est possible aujourd'hui de réaliser par son intermédiaire des tâches de nature très différentes : commander des produits, remplir une déclaration d'impôts, faire des recherches documentaires, etc.

Si l'internet est de loin la source d'informations la plus importante qui existe, il n'est pas évident d'en exploiter le contenu de manière efficace et intelligente. En effet, les pages *HTML*[1] ont deux inconvénients majeurs : d'une part, elles sont constituées d'un mélange de données et de consignes de présentation et, d'autre part, elles sont pratiquement dépourvues d'informations relatives à la structure et à la signification des données.

Afin d'améliorer les possibilités d'exploitation, il est donc intéressant d'élaborer des méthodes pour séparer les données de leur présentation, les extraire, les interpréter et retrouver leur structure sémantique invisible dans les pages *HTML*.

Les données et structures extraites deviennent une base pour des applications aussi diverses que la réingénierie de sites, l'intégration de plusieurs sites, la migration des données vers une base de données, les études statistiques, etc.

Dans le cadre de ce document, nous nous intéresserons plus particulièrement à l'extraction des données et structures de sites internet statiques en vue de leur migration vers une une base de données.

1.2 Sites statiques et sites dynamiques

1.2.1 Les sites statiques

Un site statique est composé d'un ensemble de pages *HTML* stockées sur un serveur. Le code de ces pages contient à la fois les informations (ou données) à publier et les con-

1. *HTML (HypertText Markup Language)* : format dans lequel sont produites les pages de l'internet.

signes de mise en page pour l'affichage.

Ce mélange de données et de mise en page implique de nombreuses redondances. D'une part, une donnée présente sur plusieurs pages d'un site se retrouve dans le code de chacune de ces pages. D'autre part, si la présentation de toutes les pages se fait dans un canevas général, celui-ci est également répété dans tous les fichiers source.

Les redondances rendent difficile la maintenance du site. En effet, la mise à jour d'une information ou le changement du canevas de présentation nécessite la modification de chacune des pages concernées. S'il est envisageable de gérer correctement ces évolutions lorsque le site ne compte que quelques pages, pour un site de grande taille, par contre, il est évident que des incohérences se manifesteront tôt ou tard.

1.2.2 Les sites dynamiques

Dans un site dynamique, les données et la mise en page sont séparés. Ainsi, le canevas général de présentation est enregistré dans un ou plusieurs fichiers stockés sur un serveur (par exemple, un pour l'entête, un pour le bas de page) et les données sont enregistrées dans une base de données.

Les pages *HTML* sont construites à la demande par l'intermédiaire de scripts qui réalisent plusieurs tâches :

- ils importent le canevas général;
- ils interrogent la base de données;
- ils appliquent une mise en page aux données reçues de la base de données;
- ils affichent le résultat.

De cette manière, les redondances sont supprimées. La mise à jour d'une donnée se fait une et une seule fois, et il en va de même pour le canevas de mise en page.

1.2.3 Qu'en penser ?

Au vu des ces remarques, nous pouvons nous poser la question suivante : Pourquoi élaborer des sites statiques alors que l'approche dynamique est bien plus efficace ?

La raison est simple. De nombreux outils permettent à tout un chacun de créer des pages *HTML* sans devoir nécessairement connaître les détails du langage.

Lorsqu'on se limite à un petit site personnel, cela est suffisant. Mais lorsque le nombre de pages se multiplie et que les données évoluent fréquemment, la maintenance devient difficile, voire impossible. D'où l'intérêt, alors, de migrer vers une architecture dynamique.

1.3 Notre démarche

Dans notre démarche, nous considérons les pages *HTML* d'un site statique comme des réponses à des requêtes faites sur une base de données dont on aurait perdu la structure. En ce sens, le travail à réaliser s'apparente au domaine de la rétro-ingénierie des bases de

données (voir [Hainaut 2002] pour une introduction au sujet).

Pour retrouver la structure de données perdue, nous commençons par classer les pages en catégories en fonction des informations qu'elles contiennent. Ensuite, nous analysons chaque catégorie, décelons sa structure de données sous-jacente et interprétons les données contenues. Sur la base de cette analyse, nous produisons un schéma physique de la structure et extrayons les données dans un format conforme à cette structure. Il s'ensuit un effort de conceptualisation et d'intégration des différentes structures pour obtenir un schéma conceptuel unique représentant le domaine d'application complet. A partir de ce schéma, nous menons une démarche classique d'ingénierie des bases de données pour créer une structure physique prête à accueillir les données extraites.

Brève illustration de la démarche :

Classification

Supposons un site composé de pages de deux types différents (figure 1.1). L'un décrit des départements d'une société, l'autre des employés.

Figure 1.1 - Deux pages *HTML* de types différents

Analyse

La page de gauche décrit un département. Elle donne son nom, le nom d'un responsable, un paragraphe qui décrit la mission du département, une liste d'employés et des renseignements de contact (téléphone et courrier électronique). On analyse l'autre page de la même manière.

Extraction

L'analyse nous permet de produire, d'une part, un fichier *XML* reprenant les données contenues dans la page (figure 1.2) et, d'autre part un schéma sous forme de type d'entités représentant la structure des données (figure 1.3).

```
<Département>
    <NomDep>Service technique</NomDep>
    <Responsable>François Google</Responsable>
    <Mission>
        Le <i>Service Technique</i> est responsable du bon fonctionnement de l'ensemble du matériel technique nécessaire à l'entreprise. De ce fait, il se charge d'assurer l'entretien et les réparations du matériel existant ainsi que de prévoir le remplacement du matériel amorti ou défectueux. <br>Ce service est également chargé des tâches d'entretien des bâtiments.
    </Mission>
    <Employé>Chantal Lucine: secrétaire</Employé>
    <Employé>Alfred Nimbus: responsable informatique</Employé>
    <Employé>Michel Alute: responsable commandes</Employé>
    <Contacts>
        <Téléphone>081/660033</Téléphone>
        <Courriel>clucine@company.com</Courriel>
    </Contacts>
</Département>
```

Figure 1.2 - Fichier *XML* contenant les données extraites de la page de gauche de la figure 1.1.

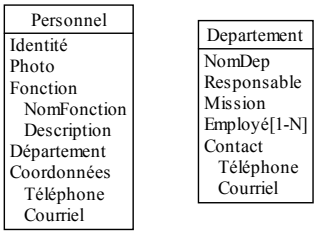


Figure 1.3 - Schéma représentant les deux structures de données dégagées de l'analyse des pages

Conceptualisation et intégration

Nous opérons des transformations sur ce schéma pour expliciter les relations entre ses éléments, intégrer les parties communes et supprimer les informations redondantes.

Dans notre cas, on peut imaginer que plusieurs personnes exercent la même fonction dans l'entreprise; nous transformons donc l'attribut *Fonction* en un type d'entités qui est relié à *Personnel* par un type d'associations.

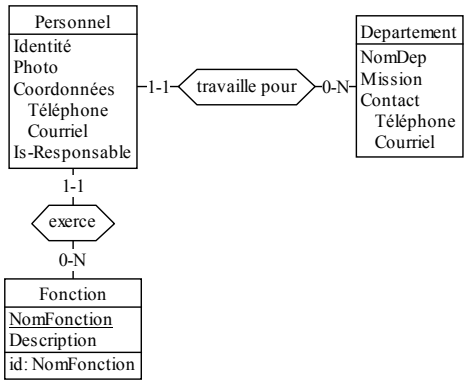


Figure 1.4 - Schéma conceptuel du site

On remarque également que l'attribut *Employé* du type d'entités *Département* correspond à l'attribut *Identité* d'une page de type *Personnel*. On transforme donc l'attribut *Employé* en un type d'entités relié à *Département* par un type d'associations, puis on le fusionne avec le type d'entités *Personnel*. Et ainsi de suite. La résultat final est le schéma conceptuel de la figure 1.4.

Conception de la base de données

A partir du schéma conceptuel, des outils nous permettent de créer une base de données et d'y enregistrer les données extraites des pages. La figure 1.5 présente les tables d'une base de données relationnelle contenant les données extraites des pages de la figure 1.1 et d'autres pages du type *Personnel*.

Fonction : Table

NomFonction	Description
Responsable informatique	Le responsable informatique est en charge de la maintenance et du
Responsable commandes	Le responsable des commandes veille à ce que tout le matériel nécessaire soit
Responsable du service technique	Le responsable du service technique répond auprès du directeur général de
Secrétaire	La secrétaire fait office de bras droit du responsable de service.

Département : Table

NomDep	Mission	Téléphone	Courriel
Service technique	Le service technique est responsable du bon fonctionnement de l'ensemble du matériel technique nécessaire à	081/660033	technique@company.com

Personnel : Table

PersID	Identité	Photo	Téléphone	Courriel	NomDep	NomFonction	Is-Responsab
1	François Google	fgoogle.jpg	081/666666	fgoogle@company.com	Service technique	Responsable du service technique	☑
2	Alfred Nimbus	animbus.jpg	081/990033	animbus@company.com	Service technique	Responsable informatique	☐
3	Chantal Lucine	clucine.jpg	081/333333	clucine@company.com	Service technique	Secrétaire	☐
4	Michel Alute	malute.jpg	081/555555	malute@company.com	Service technique	Responsable commandes	☐

Figure 1.5 - Les tables d'une base de données relationnelle pour notre exemple

Ceci fait, on peut utiliser un langage de script (par exemple *PHP*[2]) pour interroger la base de données et emballer les résultats dans une mise en page *HTML* définie à part.

1.4 Etat de l'art

Le souhait actuel du *W3C*[3] est l'avènement du *Web Sémantique*. *«The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation»*[4] [Berners-Lee et al.].

Cependant, la généralisation de cette approche n'est pas pour demain. En effet, l'internet actuel comprend des milliards de pages constituées de données noyées dans des informations de mise en page. Aussi, que ce soit pour transformet les sites actuels en sites *sémantiques* ou pour utiliser intelligemment les données enfouies, la définition de techniques

2. *PHP (Hypertext Pre-Processor)* : langage de script spécialement prévu pour le développement internet. Ce langage peut intégrer du code *HTML* ou y être intégré.
3. *W3C (World Wide Web Consortium)* : organisme qui développe des technologies visant à tirer le meilleur parti possible des potentialités de l'internet. Il se charge également de rédiger les recommandations relatives aux langages *HTML*, *XHTML* et *XML*. [W3Ca]
4. Traduction libre : *«Le Web Sémantique est une extension de l'internet tel que nous le connaissons. Les informations se voient associer une signification précise, ce qui permet une meilleure coopération entre les utilisateurs et les ordinateurs.»*

d'extraction de données revêt un grand intérêt.

Pour cette raison, l'extraction de données depuis l'internet a fait et continue de faire l'objet de nombreux travaux de recherche. [Laender et al.] en propose une classification basée sur la technique de repérage des données :

[Crescenzi et al.], [Baumgartner et al.] et [Liu et al.] se basent sur le balisage *HTML* pour générer automatiquement [Crescenzi et al.] ou semi-automatiquement (les autres) des modules d'extraction des données utiles. Chaque module d'extraction est utilisé pour extraire les données de pages dont le contenu informationnel et la structure sont homogènes.

[Soderland] et [Freitag] utilisent des techniques d'analyse du langage naturel pour définir des relations entre des phrases ou parties de phrases et en dériver des règles d'extraction.

[Adelberg] et [Leander et al.] s'appuyent sur la définition d'une structure cible pour les données à extraire. Cette structure est créée en analysant un document échantillon. En fonction de cette structurs, un algorithme définit des règles d'extraction basées sur des délimiteurs (ponctuation, constantes de texte) et parcourt d'autres documents de même type pour en extraire les données dans un format conforme à la structure cible.

[Embley et al.] s'appuyent sur les données elles mêmes. Préalablement à l'extraction, ils définissent l'ontologie du domaine. Ainsi ils décrivent les données dignes d'intérêt et leurs relations, et donnent un ensemble de constantes et mots-clés pour chaque élément du domaine. C'est sur cette base qu'ils parcourent les documents pour repérer les données à extraire. Cette méthode permet de traiter des documents hétérogènes sur le plan de la structure.

[Chung et al.] proposent une méthode mixte (balisage *HTML* et ontologies) pour intégrer des documents *HTML* homogènes sur le plan informationnel mais hétérogènes sur le plan de la structure et de la présentation. Des règles de restructuration de documents basées sur les informations structurelles et visuelles du balisage *HTML* sont utilisées pour transformer les sources en documents *XML*. Afin de donner des noms représentatifs aux éléments *XML*, l'utilisateur définit au préalable un ensemble de concepts du domaine d'application, ainsi que des exemples d'instances (mot-clés) ou des modèles d'instances pour ces concepts. Ces modèles et mots-clés sont comparés aux informations textuelles rencontrées lors de la restructuration. A partir des documents *XML*, un fichier *DTD* décrivant les structures communes est dérivé.

1.5 Contenu du mémoire

Ce mémoire s'inscrit dans le cadre du projet *RetroWeb*. Il s'agit d'un projet de recherche mené conjointement par le *CETIC*[5] et le *LIBD*[6] des Facultés Universitaires Notre-Dame de la Paix de Namur. Son objectif est de définir une méthodologie et des outils pour reconstruire la structure de données de sites internet et en extraire les données utiles. A partir de là, les possibilités d'exploitation des données sont multiples (voir section 1.1).

5. *CETIC* : Centre d'Excellence en Technologies de l'Information et de la Communication.
6. *LIBD* : Laboratoire d'Ingénierie des applications de Bases de Données.

L'objectif de ce mémoire est de prolonger un article intitulé *A tool-supported method to extract data and schema from web sites* [Estiévenart et al.]. Les auteurs y introduisent la démarche méthodologique développée dans le cadre du projet *RetroWeb*.

Ma contribution dans le cadre de ce mémoire peut-être résumée comme suit :

- définir des règles aidant l'analyste à découvrir des informations structurelles et sémantiques dans le code *HTML*.
- décrire dans le détail les étapes originales de la méthodologie;
- développer ou participer au développement de certains outils de support;
- illustrer l'application de la démarche sur un cas concret

La suite de ce document est structurée de la manière suivante :

- Le chapitre 2 présente quelques considérations sur le langage *HTML* et décrit les modèles et technologies utilisés pour mener à bien notre démarche.
- Le chapitre 3 décrit la démarche méthodologique étape par étape.
- Les outils de support pour l'application de la démarche sont décrits dans le chapitre 4.
- Le chapitre cinq illustre l'application de la méthodologie sur un cas concret.
- Pour terminer, une conclusion est proposée au chapitre 6. Elle propose quelques éléments à améliorer pour rendre l'application de la démarche plus efficace.

Chapitre 2

Le cadre de travail

Dans la première section de ce chapitre, nous décrivons brièvement le principe de départ du langage *HTML*, son évolution et les conséquences de celle-ci sur la qualité du code source. Ensuite, nous proposons quelques règles qui peuvent fournir des informations sur la structure de données sous-jacente d'un site internet.

Dans la seconde section, nous présentons les modèles et technologies que nous utiliserons pour mener à bien notre démarche.

2.1 HTML

2.1.1 Introduction

A l'origine, le langage *HTML* a été développé en tant qu'application du *SGML*[1] pour la publication internet. En ce sens, il devait fournir aux auteurs de sites un ensemble de balises destinées à structurer leurs documents.

Ainsi, des balises ont été prévues pour marquer différents niveaux de titres, des paragraphes, des tableaux, des listes, des formulaires, des liens *hypertext*,... . A chaque balise est associé un mode de présentation pour l'affichage des pages.

En évoluant, le langage a défini de nombreux éléments de formatage[2] (police de caractères, couleur,...), produisant par là un mélange d'informations de structure et de présentation dans le modèle du langage.

Ce mélange, associé à la profusion des balises, a conduit à un usage souvent détourné de certains éléments du langage. Ainsi, certains auteurs de pages choisissent certaines balises non pour leur nature structurelle, mais pour l'effet qu'elles produisent à l'écran!

Ajoutons que les utilitaires d'affichage de pages *HTML* (i.e. les navigateurs) ont une part de responsabilité dans la mauvaise qualité fréquente du code source. En effet, la pu-

1. *SGML (Standard Generalized Markup Language)* : publiée par l'*Organisation Internationale de Normalisation* (*ISO*), *SGML* est une norme qui fournit des règles pour décrire des modèles de documents en définissant des balises, leur comportement et leurs interactions. Chaque définition de modèle est spécifiée dans un document appelé *DTD (Définition de Type de Documents)*. Consultez [MARCHAL] pour une introduction au *SGML*.
2. La dernière version (*HTML 4.01*) définit une centaine de balises. [W3Cb]

blication internet étant ouverte au grand public, les développeurs de navigateurs ont prévu dans leurs produits le traitement d'un grand nombre d'erreurs de syntaxe (balises non fermées, chevauchement de balises, éléments inconnus du langage, ...). Comme le dit Benoît Marchal : *«According to some estimates, more than 50% of the code in a browser handles errors or sloppiness on the autor's part.»*[3] [Marchal]

2.1.2 Découvrir des informations dans le balisage

L'ensemble des balises décrites dans la *DTD* du langage *HTML* est fini et non extensible. Hormis pour le sous-ensemble dédié aux formulaires[4], le poids sémantique des balises est très général. Ainsi, par exemple, un couple de balise *<h1></h1>* est sensé représenter un titre de niveau 1, mais rien ne permet de savoir s'il s'agit du titre d'un livre, d'un nom de produit ou encore d'une phrase que l'auteur a simplement voulu mettre en évidence.

De plus, d'une page à l'autre, un concept de même nature peut apparaître de manière différente. Ainsi, une suite de noms peut tantôt apparaître comme éléments d'une liste, tantôt comme cellules d'un tableau, tantôt encore comme simples paragraphes de texte.

A l'opposé, des éléments de même apparence n'ont pas toujours la même sémantique ou la même importance dans la structure d'un document. Par exemple, un élément textuel formaté en gras sera tantôt une donnée mise en évidence dans un paragraphe de texte, tantôt un libellé de colonne dans un tableau, tantôt encore un titre. Difficile donc d'associer de manière automatique des informations sémantiques au balisage *HTML*.

On peut cependant remarquer des structures récurrentes dans les pages *HTML*. Il est intéressant de les répertorier et de préciser les informations tant structurelles que sémantiques qu'elles apportent. Elles donneront ainsi une base de travail lors de l'étude d'un site.

Ci-dessous, nous allons analyser les éléments suivants:

- l'adresse de la page;
- les liens *hypertext;*
- les ancres;
- les titres;
- les tableaux ;
- les listes;
- les divisions.

Pour chacun de ces élément, nous donnons sa structure technique, son usage attendu et les possibles interprétations qu'on peut lui associer.

• L'adresse de la page

Structure technique

Une adresse a la forme : *www.company.com/Personnel/ANimbus.htm*

---

3. Traduction libre : *«Selon certaines estimations, plus de 50% du code des navigateurs est chargé de traiter les erreurs et le manque de rigueur des auteurs»*.
4. La démarche décrite dans ce document s'adresse aux sites statiques. Etant donné que, par nature, les formulaires appliquent un traitement dynamique aux informations, ils ne seront pas traités dans le présent document.

On peut la décomposer en deux parties :

- l'adresse de la racine du site ou domaine du site (www.company.com)
- l'adresse relative de la page par rapport à la racine. (/Personnel/ANimbus.htm)

Usage attendu

L'adresse précise au navigateur l'endroit où est stockée la page à afficher.

Interprétations possibles

Pour pouvoir être consultée, une page doit avoir une adresse unique. Par conséquent, celle-ci est une caractéristique identifiante de la page sur l'internet.

L'adresse de la racine du site permet de délimiter l'étendue des pages à analyser. Ainsi, lorsqu'un lien (voir section •) conduit à une page d'un autre domaine, celle-ci sort du cadre de l'analyse du site de départ.

L'adresse relative (ou chemin d'accès) identifie la page au sein du site. En fonction de la structure des répertoires[5], nous pouvons considérer l'adresse relative ou le nom du fichier seul comme un attribut identifiant de la page.

Ajoutons que les fichiers contenus dans un répertoire présentent souvent le même type d'informations. Si le nom de ce répertoire est suffisamment représentatif des informations contenues dans ses fichiers, on peut le retenir pour nommer ce type d'informations.

• Les liens *hypertext*

Structure technique

Un lien *hypertext* est défini par un couple de balises *HTML* <*a*>...</*a*> qui entourent un libellé (i.e. un élément textuel) ou une image. La balise d'ouverture possède un attribut *href* dont la valeur est l'adresse d'une ancre (voir section suivante) ou d'une page *HTML*.

Usage attendu

Un lien *hypertext* est un mécanisme qui permet de passer d'une zone d'une page à une autre ou d'appeler une autre page. Pour se rendre à la cible du lien, il suffit de cliquer dessus.

Interprétations possibles

On rencontre différents types de liens :

- les liens de navigation;
- les liens externes;
- les liens sémantiques.

Les liens de navigation

Un lien de navigation aide le visiteur du site à se déplacer facilement à l'intérieur d'une page ou entre les pages du site (retour au début de la page, retour à la page d'accueil,

5. Selon qu'un même nom de fichier se retrouve dans un ou plusieurs répertoires.

etc.).

Ces liens ne font pas partie du contenu informationnel du site. Néanmoins, certains d'entre eux peuvent donner des indications sur la structure du site ou sur les types d'informations qui y sont présentées. Notons par exemple l'usage fréquent d'une colonne de liens sur la gauche des pages. Souvent, chacun de ces liens mène à un type de pages différent ou à un point d'entrée vers une partie homogène du site.

Les liens externes

Comme son nom l'indique, un lien externe mène vers une page d'un autre site. Par conséquent, les informations contenues dans la page cible sortent du cadre de l'analyse. Cependant, si l'objectif de l'analyse est de reconstruire le site, il convient de conserver l'adresse référencée afin de pouvoir la réutiliser.

Les liens sémantiques

Un lien sémantique traduit une relation forte entre l'appelant (lien) et l'appelé (ancre ou page). Supposons que des pages présentent les employés d'une société. Chacune de ces pages contient un lien dont la cible est une page qui décrit un département. Le lien représente dans ce cas une propriété importante du domaine d'application : *Chaque employé est attaché à un département*.

Notons qu'il arrive que plusieurs liens dans une même page référencent la même adresse. On peut, dans ce cas, n'en considérer qu'un lors de l'analyse.

Lorsque le contenu du lien est un libellé, ce dernier peut être de deux natures : une constante ou une donnée variable. Dans l'exemple pris ci-dessus, nous pourrions avoir respectivement *lien vers la page de mon département* (constante) et *Service Technique* (donnée variable). Dans le cas d'une donnée variable, sa valeur est généralement présente également dans la page référencée (redondance).

• Les ancres

Structure technique

Une ancre est définie par une balise *HTML* <*a*/> (ou un couple <*a*>...</*a*>) qui possède un attribut *name*.

Une page peut contenir un nombre quelconque d'ancres, mais la valeur de l'attribut *name* doit être différente pour chacune d'entre elles.

Usage attendu

Une ancre est un mécanisme qui localise la cible d'un ou plusieurs liens internes ou externes à la page dans laquelle elle est définie (cette propriété n'est pas visible dans la fenêtre d'un navigateur).

Interprétations possibles

En raison de la contrainte d'unicité imposée sur la valeur de l'attribut *name*, cette valeur, associée à l'adresse relative de la page, identifie au sein du site l'information localisée par l'ancre

On rencontre deux types d'ancres :

- les ancres de navigation : mécanisme inverse d'un lien de navigation;
- les ancres sémantiques : mécanisme inverse d'un lien sémantique.

• Les titres

Structure technique

HTML propose six niveaux de titres, respectivement définis par les couples de balises *<h1>...</h1>* à *<h6>...</h6>* qui entourent un élément textuel.

Usage attendu

Les titres sont destinés à donner une structure hiérarchique aux informations présentées dans une page.

Interprétations possibles

En plus des indications structurelles (hiérarchie), les titres peuvent fournir des informations sémantiques sur les données grâce à leur contenu textuel. En effet, lorsque celui-ci est une constante, elle est généralement représentative des informations présentées sous le titre. On peut alors raisonnablement utiliser tout ou partie de la constante pour nommer le concept décrit.

Par contre, lorsque le contenu textuel est une donnée variable, il faut généralement consulter les informations affichées sous le titre pour découvrir la nature du concept décrit.

Notons enfin que les balises de titre font parfois l'objet d'un usage détourné. Ainsi, certains les utilisent simplement pour mettre du texte en évidence. Dans ce cas, il est difficile d'en tirer des informations.

• Les tableaux

Structure technique

Les tableaux *HTML* sont représentés par une série de balises dont voici les plus fréquentes :

- le couple *<table>...</table>* délimite le tableau;
- le couple *<tr>...</tr>* définit une ligne dans le tableau;
- le couple *<td>...</td>* définit une cellule dans une ligne de tableau.

Usage attendu

Les tableaux sont destinés à représenter les différentes instances d'un concept répétitif (lignes du tableau) et les différents éléments d'un concept composé (colonnes du tableau).

Interprétations possibles

On rencontre deux types de tableaux dans les pages *HTML :*

- les tableaux d'alignement;
- les tableaux de données.

Les tableaux d'alignement

Les tableaux *HTML* sont très souvent utilisés à des fins de mise en page. Ce type de tableaux divise toute la page ou une partie de celle-ci en zones indépendantes. Cette méthode permet de gérer facilement l'alignement des éléments.

On voit ainsi fréquemment des pages divisées en deux colonnes. La première contient des liens de navigation, la seconde est divisée en trois cellules (en-tête, contenu principal et pied-de-page).

Ces tableaux ne fournissent aucune information sur la structure de données des pages.

Les tableaux de données

Comme leur nom l'indique, les tableaux de données se caractérisent par un réel contenu informationnel. L'interprétation qu'on en donne dépend de leur structure.

Le modèle libelle : valeur

La figure 2.1 présente un exemple du modèle *libellé : valeur*.

Figure 2.1 - Exemple du modèle *libellé : valeur*

On remarque un tableau de deux colonnes précédé d'un titre. Les cellules de la première colonne contiennent le nom des sous-éléments du concept composé représenté par le titre. Les cellules de la seconde colonne contiennent les valeurs affectées aux sous-concepts.

Les concepts multivalués (ou répétitifs)

Lorsque les valeurs des cellules dans les lignes successives du tableau se ressemblent, il y a de fortes chances qu'elles représentent les différentes instances d'un concept multivalué (usage normal d'un tableau).

Si les cellules de la première ligne ont un formatage particulier, leur contenu est généralement un libellé de colonne. Dans ce cas, il peut être retenu pour nommer le concept représenté par la colonne.

Notons qu'il n'est pas toujours trivial de déceler un tableau représentant les instances d'un concept multivalué. En effet, les cellules de tableaux peuvent être complexes et contenir à leur tour une hiérarchie de sous-concepts.

• Les listes

Structure technique

HTML propose deux types de listes : les listes numérotées (<*ol*>...</*ol*>) et les listes à puces (<*ul*>...<*ul*>). Chaque élément d'une liste est précédé d'une balise <*li*>. Une liste peut-être insérée au sein d'une autre liste.

Usage attendu

Les listes sont destinées à représenter les différentes instances d'un concept répétitif (comme les tableaux d'une seule colonne).

Interprétations possibles

Lorsqu'une liste est précédée d'une constante textuelle mise en évidence (titre, gras, italique), celle-ci peut souvent être reprise pour nommer le concept.

Les listes font parfois l'objet d'un usage détourné. Ainsi, certains les utilisent pour mettre un morceau de texte en retrait par rapport au texte principal (nota bene) ou encore pour représenter un titre.

• Les divisions

Structure technique

Les divisions sont représentées par les balises <*div*>...</*div*> (division inter-paragraphe) et <*span*>...</*span*> (division intra-paragraphe). Les divisions peuvent être multi-niveaux.

Usage attendu

Les divisions inter-paragraphe servent à regrouper des éléments d'une page *HTML* en blocs homogènes superposés. Elles permettent ainsi de donner une structure claire à la page.

Les divisions intra-paragraphe servent à isoler une partie d'un élément textuel afin de lui associer un formatage particulier.

Interprétations possibles

Les divisions inter-paragraphe sont caractéristiques des pages bien structurées.

Une application typique des divisions consiste à regrouper le titre d'un paragraphe et son contenu.

L'ensemble des divisions matérialise la structure globale choisie par l'auteur pour publier sa page. Chaque division contient alors un concept de la structure de données.

Notons toutefois que les divisions sont parfois utilisées à des pures fins de mise en page (couleur de fond, alignement).

Destinées principalement au formatage, les divisions intra-paragraphes sont moins intéressantes. Néanmoins, elles permettent parfois de localiser précisément une donnée utile perdue au milieu d'un paragraphe de texte pauvre en contenu informationnel.

2.2 Modèles et technologies utilisés

2.2.1 *XML* : *eXtensible Markup Language*

Comme nous l'avons vu dans la section précédente, les sources *HTML* sont pauvres en sémantique, mais peuvent fournir des informations sur la structure des données.

Si nous voulons parvenir à définir une structure de données représentative d'un site internet, il faut absolument ajouter de la sémantique aux informations structurelles.

On pourrait donc imaginer d'ajouter des éléments (balises et attributs) à caractère sémantique aux sources *HTML*. Cependant, à chaque version de la norme *HTML*, les balises autorisées sont définies et figées. Ce langage ne permet pas à l'utilisateur d'ajouter ses propres balises.

Il nous faut donc trouver un autre moyen pour représenter les informations sémantiques. Nous avons choisi *XML* pour les raisons suivantes :

XML est extensible

XML est également un langage de balisage issu de *SGML*, mais, à l'inverse de *HTML*, il est extensible. La norme ne définit aucune balise. Elle donne aux utilisateurs les règles pour créer leurs propres jeux de balises.

Etant donné qu'elles ne sont pas prédéfinies, les balises d'un document *XML* n'ont pas de format d'affichage par défaut. Contrairement à *HTML*, il n'y a pas dans *XML* de mélange d'informations de structure et de présentation.

XML impose une syntaxe stricte

XML impose une syntaxe plus stricte que *HTML*.

Par exemple, tout élément *XML* doit être fermé, contrairement à certains éléments *HTML* qui, par définition, ont un contenu vide (<*meta*>, <*br*>, ...).

Autre exemple, la définition du langage *HTML* recommande d'entourer les valeurs d'attributs par des guillemets ou apostrophes, mais permet dans certains cas de les omettre. En *XML* la règle est obligatoire dans tous les cas.

Le respect d'une syntaxe stricte permet de développer des outils plus légers et plus rapides. On peut analyser des documents sans devoir gérer les erreurs syntaxiques.

La définition d'une syntaxe stricte a donné naissance à la notion de *document bien-formé au sens XML*. Tout document qui respecte scrupuleusement la norme peut être qualifié de bien formé. La description complète et normative de la syntaxe *XML* peut être consultée sur le site du *W3C* [W3Cc].

XML offre un ensemble de normes et outils annexes

Le développement de *XML* s'est accompagné de la définition d'un ensemble de normes et outils annexes[6] qui poursuivent chacun un objectif propre. La section suivante présente brièvement ceux que nous utiliserons dans notre démarche.

6. Connus sous le nom de *companion standards*.

La description exhaustive de ces *companion standards* sort du cadre de ce document. Aussi nous nous contentons de donner quelques considérations générales. Nous renvoyons le lecteur désireux d'approfondir le sujet à [Marchal] pour une présentation plus détaillée de *XML* et des *companion standards* ou au site du *W3C* [W3Ca] pour les adeptes des textes normatifs.

2.2.2 Les *companion standards*

- **Les parsers**

Un *parser* est l'utilitaire de base pour le traitement d'un document *XML*. Il sert d'intermédiaire entre le document et le programme d'application. Il permet à ce dernier de parcourir le document sur la base de l'arbrorescence formée par les éléments, attributs et données textuelles sans devoir se soucier de la syntaxe *XML*.

- **Les espaces de noms**

Le fait que *XML* soit extensible permet à tout utilisateur de définir ses propres langages. Par conséquent, il est inévitable que des mêmes noms de balises soient utilisés par des personnes différentes pour représenter des choses différentes. Un conflit survient lorsqu'on souhaite utiliser des éléments de même nom venant de langages différents.

Afin d'éviter ces conflits, on associe un espace de noms à toutes les balises d'un même langage. L'espace de nom est représenté par une *URI*[7], qui, par construction, peut être rendue unique moyennant un minimum de discipline.

Lorsqu'un document utilise des éléments provenant de différents espaces de noms, on déclare ceux-ci et on leur associe un préfixe (sauf pour l'éventuel espace de noms par défaut). Dans le document, chaque balise contient le préfixe adéquat et le nom de l'élément séparés par « : ». Chaque élément associé à un tel préfixe est dit *qualifié*. Les éléments appartenant à l'espace de noms par défaut sont, eux, dépourvus de préfixe.

Dans notre méthodologie, nous définirons un ensemble de balises que nous associerons à un espace de noms. Ces balises serviront à créer un document décrivant la structure et la sémantique des données contenues dans un type de pages *HTML*.

- **Le modèle XML Schema et la notion de document valide**

La norme *XML Schema* [W3Cd] et [W3Ce] permet de définir des modèles de documents *XML*[8]. Lorsqu'on associe un *XML Schema* à un document, ce dernier est appelé une *instance* du modèle. Il est dit *valide* s'il est bien formé et est conforme au modèle associé.

Un *XML Schema* est un document *XML* qui décrit les éléments qui peuvent apparaître dans ses instances, ainsi que la hiérarchie que ceux-ci doivent respecter. Il précise les éventuels aspects répétitifs ou facultatifs des éléments ainsi que leur type.

La norme prévoit un certain nombre de types de données pré-définis (*string*, *integer*, *date*,

7. *URI (Uniform Resource Identifier)* : chaîne de caractères qui identifie des ressources sur l'internet (documents, images, boîte aux lettres électroniques, fichiers, ...)
8. *XML Schema* est en ce sens un concurrent du modèle des *DTD*.

anyUri, etc.) et permet de définir des types simples ou complexes.

Les types simples

Les types simples représentent des éléments *XML* qui ne contiennent que des données textuelles. On peut les définir de trois manières :

- par restriction d'un type prédéfini ou d'un autre type simple;
- par l'union de deux ou plusieurs types simples ou prédéfinis;
- en tant que liste d'éléments d'un type simple ou prédéfini.

Les types complexes

Les types complexes décrivent tous les autres cas (présence de sous-éléments et/ou d'attributs et/ou de données textuelles) et sont caractérisés par un type de contenu : vide, simple, complexe ou mixte. Voici ce que renferme chaque type de contenu :

- contenu vide : uniquement attributs;
- contenu simple : données textuelles et attributs;
- contenu complexe : sous-éléments (et éventuellement attributs);
- contenu mixte : sous-éléments, données textuelles, et éventuellement attributs.

On associe en outre un modèle de contenu aux deux derniers types. Ce modèle précise la façon dont les sous-éléments doivent apparaître dans les instances (selon un ordre précis ou non).

Ajoutons encore qu'il est possible de définir des structures différentes pour un même élément au sein de sa déclaration, laissant ainsi à l'auteur le choix lors de la rédaction d'instances.

Pour qu'un document soit valide, il faut que tous ses éléments constitutifs soient définis dans le *XML Schema* associé et respectent sa hiérarchie. La norme prévoit toutefois une exception à cette règle. Elle autorise l'utilisation d'un élément joker (ou *wild-card*) qui permet de valider des éléments non définis dans le *XML Schema*. La seule contrainte est que la section cachée par la *wild-card* soit bien formée au sens *XML*. Ce mécanisme est notamment utilisé pour insérer du code *HTML* dans un document *XML*.

Nous utiliserons le modèle *XML Schema* pour représenter la structure physique des données de chaque type de pages.

• XSL

Comme nous l'avons dit plus haut, l'objectif de *XML* est de décrire le contenu et la structure d'un document sans se soucier de sa présentation. Cette indépendance entre le fond et la forme permet d'utiliser les documents pour de multiples applications différentes (publication papier, publication web, traitement de données, réquisitions, ...)

XSL (*eXtensible Stylesheet Language*) regroupe une série de normes visant à transformer et/ou présenter des documents *XML*. Ainsi, on peut éditer les données dans des formats divers (*HTML*, *PDF*, *WML*, ...) et également produire de nouveaux documents (nouvelle structure pour les données, sélection d'une partie seulement d'un document, génération

d'un *XML Schema* pour un document de données, etc.).

Parmi les normes composant *XSL*, nous nous intéressons plus particulièrement à *XPath* et *XSLT*.

XPath

XPath [W3Cf] est un langage dont le but est d'opérer des sélections dans des documents *XML*. Il considère le document comme un arbre composé de noeuds de trois types : les éléments, les attributs et les données textuelles.

La sélection des noeuds s'opère sur la base d'un *chemin* qui est soit absolu (chemin complet depuis la racine du document) soit relatif (par rapport à un noeud de référence). Les *chemins* peuvent être utilisés tels quels pour sélectionner un ensemble de noeuds ou être utilisés dans des expressions comme arguments de fonctions prédéfinies.

Un chemin *XPath* est divisée en trois parties : l'axe, le noeud et le prédicat.

L'axe

L'axe détermine la partie du document qu'il faut considérer par rapport au noeud de référence ou à la racine (fils, descendants, père, ancêtres, frères suivants, ...). Si l'axe n'est pas explicitement donné, le *chemin* considère par défaut l'axe des fils.

Le noeud

Le noeud précise le nom du/des noeud(s) à prendre en considération dans l'ensemble des noeuds de l'axe.

Le prédicat

Le prédicat est facultatif. Il permet de préciser une ou plusieurs conditions que les noeuds doivent remplir pour être sélectionnés (par exemple la présence d'un attribut précis).

XSLT

XSLT (*XSL Transformations*) [W3Cg] est un langage qui permet de transformer un document *XML* en un autre document *XML*.

Un document *XSLT* définit des règles de transformation (appelées *templates*) à appliquer sur un arbre source construit à partir d'un document *XML*. Chaque règle précise les traitements à effectuer sur les noeuds sélectionnés au moyen d'un chemin ou d'une expression *XPath*.

De l'appel d'une règle résulte un morceau de l'arbre cible. Lorsque l'arbre source a été entièrement parcouru conformément aux critères de sélection, l'arbre cible généré est traduit en un nouveau document *XML*.

Les documents source et cible peuvent avoir une structure totalement différente : réorganisation des éléments, suppression d'éléments, d'attributs ou de noeuds de textes, emballage des éléments dans du code *HTML*,...

Nous utiliserons *XSLT* pour générer automatiquement un *XML Schema* au départ du document descriptif de la structure et de la sémantique d'un type de pages.

2.2.3 Le modèle *GER*

Comme le dit Christine Delcroix dans son document *Conventions de représentation d'un DTD dans le modèle GER*[9] [Delcroix], *«Le modèle Entité Association Etendu permet la représentation de structures de données issues de multiples paradigmes (relationnel, système de fichiers, orienté objet, hiérarchique,...) à différents niveaux d'abstraction (physique, logique ou conceptuel). On peut donc envisager la représentation de la structure de données XML dans ce même modèle.»*

Le document cité définit avec précision les règles de traduction d'une *DTD* dans le modèle *GER*. Un autre document [François] s'est employé à adapter le premier pour le modèle *XML Schema*.

La figure 2.2 présente les principaux objets du modèle *GER*. Nous les décrivons brièvement ci-dessous. Cet exemple s'inspire de [Hainaut et al.].

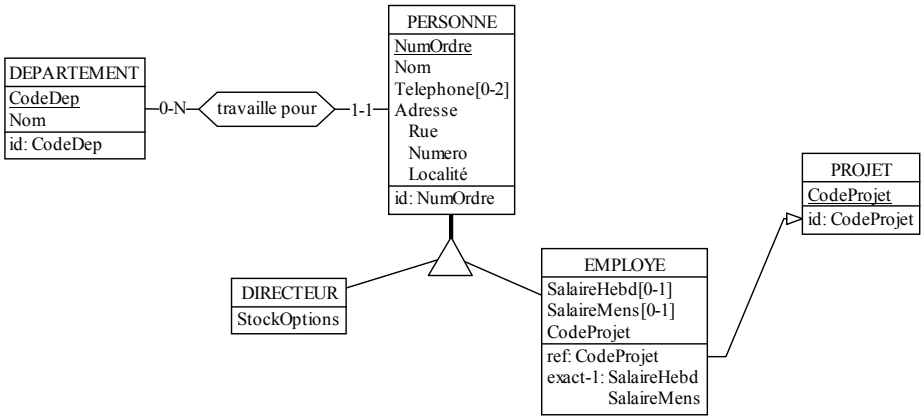


Figure 2.2 - Schéma présentant les principaux objets du modèle *GER*.

- Les principaux concepts du domaine analysé sont représentés par des *types d'entités* : *PERSONNE*, *DEPARTEMENT*, *EMPLOYE*, *DIRECTEUR* et *PROJET*.
- Les caractéristiques de ces concepts sont matérialisés par des *attributs*. Par exemple : *CodeDep* et *Nom* sont les attributs du type d'entités *DEPARTEMENT*.
- *NumOrdre* est un attribut identifiant du type d'entités *PERSONNE*. Cela signifie que deux instances de ce type d'entités ne peuvent avoir la même valeur pour l'attribut *NumOrdre*.
- Un attribut peut être facultatif et/ou répétitif. Dans l'exemple, c'est le cas de l'attribut *Telephone* du type d'entités *PERSONNE*. Chaque personne peut avoir de 0 à 2 numéros de téléphone.
- Un attribut peut être soit atomique (par exemple *Nom*), soit composé (par exemple *Adresse*).
- Les types d'entités *DIRECTEUR* et *EMPLOYE* sont des spécialisations (ou sous-types) du type d'entités *PERSONNE*. Cela signifie qu'elles possèdent à la fois les attributs de leur sur-type et des attributs propres.
- Les attributs *SalaireHebd* et *SalaireMens* du type d'entités *EMPLOYE* sont soumis à une

9. *Generic Entity Relationship Model* : en français *Modèle Entité Association Etendu*.

contrainte d'existence (ici *exact-1*). Celle-ci signifie qu'un employé a soit un salaire hebdomadaire, soit un salaire mensuel, mais pas les deux.

- L'attribut *CodeProjet* du type d'entités *EMPLOYE* est soumis à une contrainte référentielle (*ref*) qui le relie au type d'entités *PROJET*. Cela signifie que tout employé est affecté à un projet. Le domaine de valeur de l'attribut *CodeProjet* du type d'entités *EMPLOYE* est inclus dans le domaine de valeur de l'attribut *CodeProjet* du type d'entités *PROJET*.
- Une relation entre les types d'entités *DEPARTEMENT* et *PERSONNE* est matérialisée par un *type d'associations* (*travaille pour*). Chacun des deux types d'entités joue un *rôle* dans cette relation. Les rôles sont caractérisés par une cardinalité minimum et maximum. Dans notre exemple, le rôle joué par le type d'entités *PERSONNE* est de cardinalité [1-1] et celui joué par le type d'entités *DEPARTEMENT* est de cardinalité [0-N]. Cela signifie, d'une part, que toute personne travaille pour un département et, d'autre part, qu'un département employe un nombre indéterminé de personnes.

Dans notre démarche, nous utilisons le modèle *GER* pour représenter des fichiers *XML Schema* et les transformer en un schéma conceptuel.

Une description plus détaillée du modèle *GER* peut-être trouvée dans [Hainaut 1995] au chapitre 4.

Chapitre 3

La démarche méthodologique

Dans ce chapitre, nous décrivons en détail notre démarche méthodologique.

La première section introduit le sujet. La deuxième section propose une vue schématique de l'ensemble. Les sections 3 à 9 décrivent chacune des étapes.

Afin d'illustrer certains points, nous prenons pour exemple le cas d'un site qui décrit les départements et les membres du personnel d'une société.

3.1 Introduction

L'objectif de notre démarche méthodologique est d'extraire les données utiles d'un site internet statique[1] pour les importer dans une base de données. Les données seront alors accessibles de manière dynamique et permettront de reconstruire les pages du site à la demande. On peut diviser la démarche en deux grandes phases :

- la rétro-ingénierie;
- la réingénierie.

3.1.1 La rétro-ingénierie

La démarche de rétro-ingénierie consiste, d'une part, à découvrir la structure de données sous-jacente du site et, d'autre part à extraire les données utiles disséminées dans les pages.

Pour construire la structure de données, il faut identifier les différents types d'informations présentées et les relations entre ces informations.

Habituellement, le contenu d'une page *HTML* est assez homogène. Les informations qui y sont présentées sont les détails d'un concept plus général représenté par la page entière. Par exemple, la page personnelle d'un employé d'une société ne contient que les informations qui concernent cet employé.

Pour cette raison, nous choisissons la page comme unité d'analyse.

1. Raison pour laquelle ce document ne traite pas les formulaires *HTML*

Par conséquent, nous commençons par répertorier les différents types de pages du site et construisons la structure de données de chaque type. Ensuite nous exprimons les relations entre les types de pages dans un schéma conceptuel représentatif du site complet.

L'extraction des données se fait par type de pages sur la base de la structure de données dégagée. Les données sont produites dans des fichiers *XML*.

3.1.2 La réingénierie

La réingénierie consiste à construire les structures physiques d'une base de données au départ du schéma conceptuel du site. Les structures construites dépendent du système de gestion de bases de données (ou *SGBD*) choisi.

On procède ensuite à la migration des données extraites vers la nouvelle base de données.

La procédure de réingénierie correspond à une démarche classique de conception logique et physique dans le domaine de l'ingénierie des bases de données. Ce sujet ayant fait l'objet de nombreux travaux et publications au sein du *LIBD*, nous n'en donnerons qu'une définition générale dans ce document. Le lecteur désireux d'approfondir le sujet peut consulter [Hainaut 2001] chapitres 8 à 11.

3.2 Vue schématique de la méthodologie

Nous énonçons ci-dessous les différentes étapes de notre méthodologie et les rassemblons dans un schéma (figure 3.1). Les sections suivantes de ce chapitre décrivent ces étapes en détail.

1. **Préparation :** nous répertorions l'ensemble des pages du site et les transformons en documents bien formés au sens *XML* (section 3.3).
2. **Classification des pages :** nous regroupons les pages du site en *types de pages* en fonction de leur contenu informationnel (section 3.4).
3. **Enrichissement sémantique :** pour chaque type de pages, nous répertorions, nommons et hiérarchisons les concepts qui le composent dans un fichier *XML* appelé description *Meta*[2] (section 3.5).
4. **Extraction :** sur la base de la description *Meta*, nous extrayons un fichier *XML* qui contient l'ensemble des données localisées dans les pages *HTML*. La description *Meta* nous permet également de générer un fichier *XML Schema* qui décrit la structure de données. (section 3.6).
5. **Intégration et conceptualisation :** nous transformons et intégrons les différents *XML Schemas* pour obtenir un schéma conceptuel unique. (section 3.7).
6. **Conception :** nous menons un processus classique d'ingénierie des bases de données afin de créer une structure physique qui accueillera les données. (section 3.8).
7. **Migration :** nous enregistrons les données extraites dans la base de données. (section 3.9).

2. *Meta* : vient du terme *meta-langage* : langage qui permet de définier d'autres langages ou de modifier un langage existant.

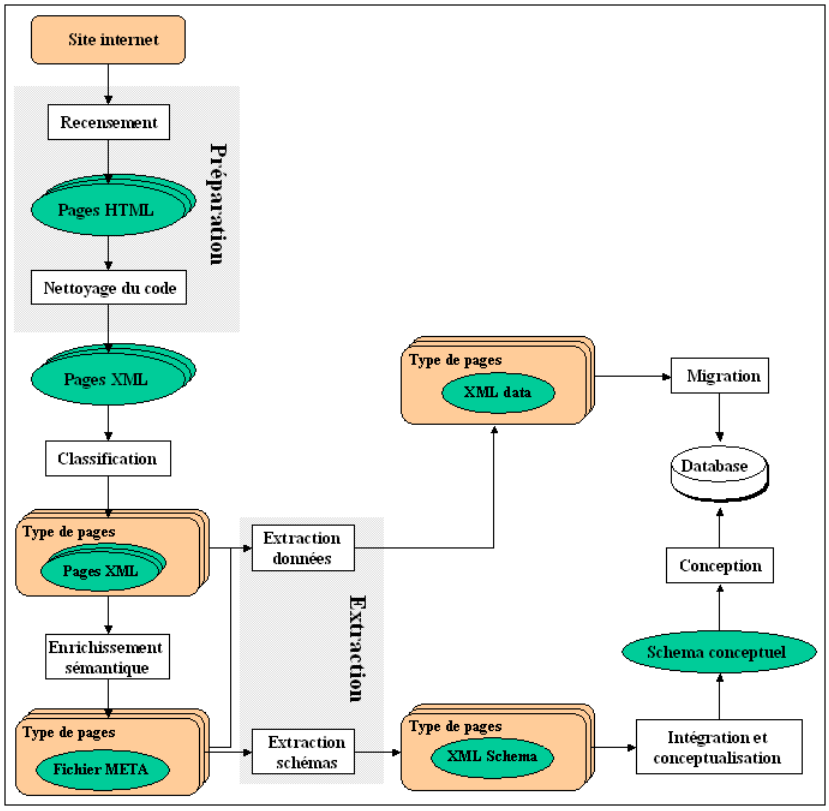


Figure 3.1 - Schéma général de la méthodologie

3.3 Préparation

3.3.1 Recensement des pages

Lors de cette étape, nous recensons l'ensemble des pages dans un tableau dont chaque ligne contient l'adresse relative d'une page par rapport à la racine du site. (figure 3.2)

Adresse racine = http://www.company.com
Accueil.htm
ListeDep.htm
Departements/Direction.htm
Departements/Technique.htm
Personnel/ANimbus.htm
.....

Figure 3.2 - Exemple de tableau de recensement des pages

3.3.2 Nettoyage du code *HTML*

Comme nous l'avons noté au chapitre précédent, les navigateurs internet permettent aux concepteurs de créer leurs pages *HTML* sans devoir faire preuve d'une rigueur absolue. Une grande partie des composants de ces navigateurs sont consacrés au traitement des erreurs dans les fichiers sources afin d'afficher ceux-ci d'une manière aussi proche que possible des souhaits supposés de l'auteur.

C'est sur ce type de sources que nous allons appliquer notre méthodologie. Nous souhaitons ne pas être contraints de gérer tout au long du processus les erreurs de syntaxe et de structure. Aussi décidons-nous de régler ce problème une fois pour toutes en effectuant un pré-traitement du code source avant de démarrer notre processus proprement dit.

Lors de ce pré-traitement, nous ne nous bornons pas à corriger les erreurs, mais nous en profitons pour faire des sources *HTML* des fichiers bien formés au sens *XML*. Ce travail permettra l'utilisation d'outils existants pour le traitement de fichiers *XML*, notamment les *parsers* et *XSLT* (voir chapitre 2, section 2.2.2).

3.4 Classification des pages

La classification des pages consiste à regrouper en diverses catégories l'ensemble des pages du site internet analysé. Le regroupement se fait en fonction des types d'informations contenues dans les pages et de la façon dont sont présentées ces informations. Dans la suite du document ces catégories sont appelées *types de pages*.

Afin de mener à bien cette classification, nous analysons le site sous trois angles :

1. l'arborescence de navigation;
2. la structure des chemins d'accès aux fichiers;
3. l'analyse visuelle des pages du site.

3.4.1 L'arborescence de navigation

L'arborescence de navigation nous permet, au départ d'un point d'entrée[3] généralement pauvre, voire vide en termes de données, de voyager vers les pages à contenu pertinent.

Généralement, en plus d'une présentation générale, une page d'accueil propose des liens vers d'autres pages. A leur tour, les pages référencées peuvent contenir des liens vers d'autres pages et ainsi de suite. La figure 3.3 illustre l'arborescence de navigation pour notre exemple.

3. Souvent appelé *page d'accueil*

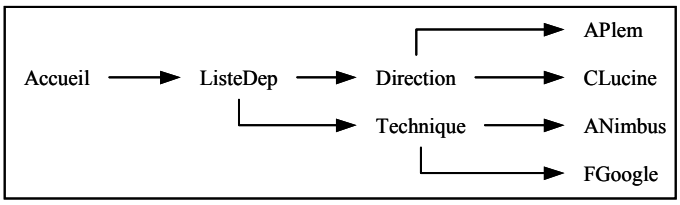


Figure 3.3 - Exemple d'arborescence de navigation

3.4.2 La structure des chemins d'accès aux fichiers

La manière dont sont structurés les répertoires contenant les pages sources donnent de bonnes indications sur les types de pages. En effet, dans un souci de maintenance aisée, le concepteur des pages regroupe généralement toutes les pages d'un type dans un même répertoire. Ainsi, dans notre exemple, les pages décrivant les membres du personnel se trouvent toutes dans le même répertoire.

La consultation du tableau de recensement des pages (voir figure 3.2.) permet d'analyser la structure des chemins d'accès aux fichiers.

En joignant les informations recueillies des deux angles d'analyse, on dégage une première classification des pages. Dans notre exemple, nous identifions deux types de pages : *Departement* et *Personnel.*

Nous ajoutons au tableau de recensement des pages une colonne qui mentionne le type associé à chaque page (figure 3.4). Celles qui n'ont pu être classifiées dans un type précis (la page d'accueil en fait généralement partie) se voient attribuer le type générique *page orpheline*.

Adresse racine = http://www.company.com	**Type de pages**
Accueil.htm	Orpheline
ListeDep.htm	Orpheline
Departements/Direction.htm	Departement
Departements/Technique.htm	Departement
Personnel/ANimbus.htm	Personnel
.....	...

Figure 3.4 - Tableau associant chaque page à son type

3.4.3 Analyse visuelle des pages

Nous confrontons la classification à une analyse visuelle des pages du site afin de vérifier la correspondance sémantique et structurelle entre les pages classées dans une même catégorie. Nous pouvons ainsi confirmer nos choix ou éventuellement les adapter.

Cette démarche conduit tantôt à diviser l'un ou l'autre type de pages en plusieurs nouveaux types, tantôt à fusionner différents types de pages.

3.5 Enrichissement sémantique

Comme nous l'avons déjà signalé, le langage *HTML* est très pauvre en sémantique. Les informations qu'on peut en tirer sont insuffisantes pour retrouver la structure sémantique d'un site. Il nous faut donc ajouter au code *HTML* des informations structurelles et sémantiques.

Pour ce faire, nous allons, pour chaque type de pages, définir une grammaire que nous appelons *description Meta*. Elle prend la forme d'un fichier *XML*.

La description *Meta* répertorie, nomme et hiérarchise l'ensemble des concepts identifiés dans un type de pages. A cette fin, elle utilise un formalisme composé d'éléments *XML*. Elle intègre également le balisage *HTML* du fichier analysé afin de localiser précisément les concepts.

Le fichier créé sert de modèle pour extraire les données des instances du type de pages.

Ci-dessous, nous décrivons le formalisme du *Meta* en introduisant l'un après l'autre chaque élément et attribut *XML* utilisés.

Pour illustrer notre propos, nous prenons pour exemple le type de pages *Personnel* dont la figure 3.5 présente une instance.

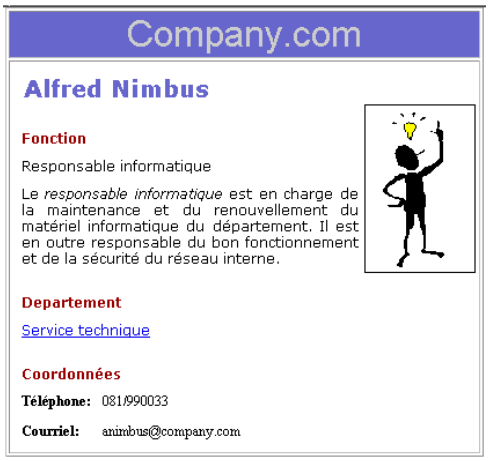


Figure 3.5 - La page «ANimbus.htm» (exemple du type de pages *Personnel*)

Des représentations graphiques des structures crées sont proposées tout au long de cette description[4]. La figure 3.6 donne la signification des éléments graphiques utilisés et la figure 3.7 présente l'arbre formé par les éléments *HTML* de la page prise en exemple.

4. Pour les amateurs, le code *XML* correspondant aux graphiques est disponible à l'Annexe A, section A.1.

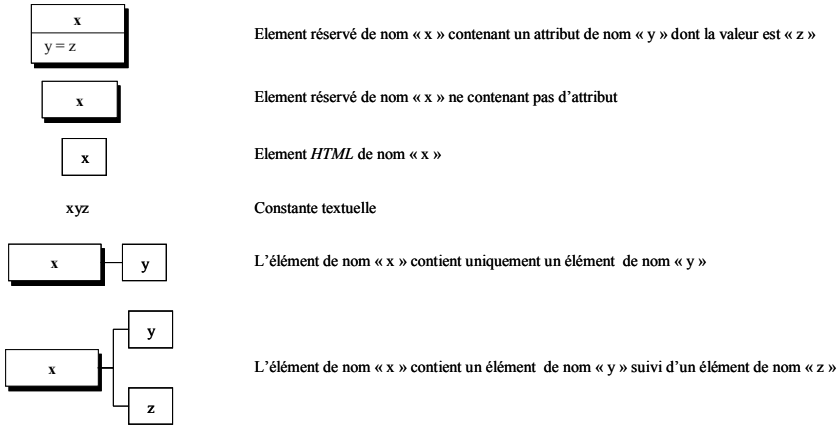


Figure 3.6 - Description des éléments graphiques utilisés.

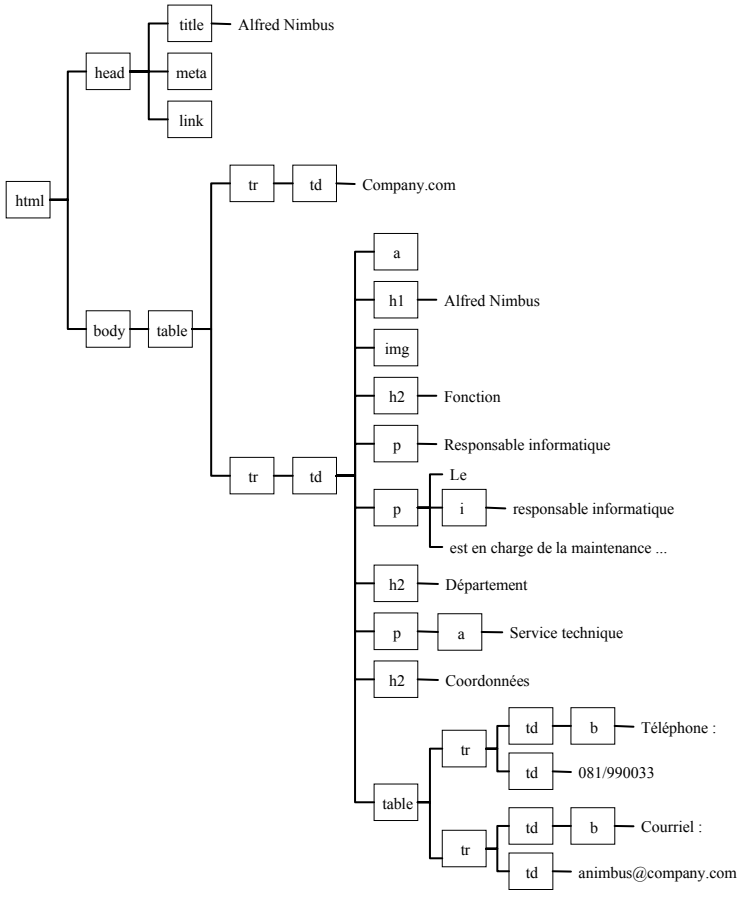


Figure 3.7 - Arbre *HTML* correspondant à la figure 3.5

3.5.1 Le formalisme du fichier *Meta*

- **La racine du fichier *Meta* : <*meta:htmldescription*>**

Pour qu'un fichier *XML* soit bien formé, sa racine doit être unique. Par convention, elle est identique dans tous les fichiers *Meta* et prend la forme de l'élément *XML* <*meta:htmldescription*>. Elle contient une déclaration d'espace de noms qui prend la forme d'un attribut (figure 3.8). Cet espace de noms est utilisé pour qualifier tous les éléments propres à notre formalisme.

Figure 3.8 - Représentation graphique de la racine d'un fichier *Meta*.

- **La représentation d'un concept : <*meta:element*>**

L'élément *XML* <*meta:element*> sert, d'une part, à définir chacun des concepts identifiés dans un type de pages et, d'autre part, à faire référence à un concept au sein d'un autre.

Afin de différencier ces deux usages, cet élément est toujours accompagné d'un et un seul des deux attributs suivants :

- *name* : pour la déclaration d'un concept;
- *ref* : pour référencer un concept au sein d'un autre.

La déclaration d'un concept : l'attribut *name*

Chaque concept identifié dans un type de pages est représenté par un <*meta:element*> pourvu d'un attribut *name* qui permet de lui donner un nom représentatif. On le déclare comme fils direct de la racine du document. Ainsi, l'ensemble des fils directs de la racine constitue la liste de tous les concepts présents dans la page.

Dans notre exemple, un coup d'oeil à la page de référence nous permet de découvrir les concepts suivants : *Identite, Photo, Fonction, DescriptionFonction, Departement, Coordonnees, Telephone* et *Courriel* (voir figure 3.5).

Notons un cas particulier dans la déclaration des concepts : on déclare comme premier fils de la racine un concept principal qui représente la globalité des informations contenues dans la page. Dans notre cas, le premier élément de la liste est donc le concept *Personnel*. La figure 3.9 présente la liste des concepts.

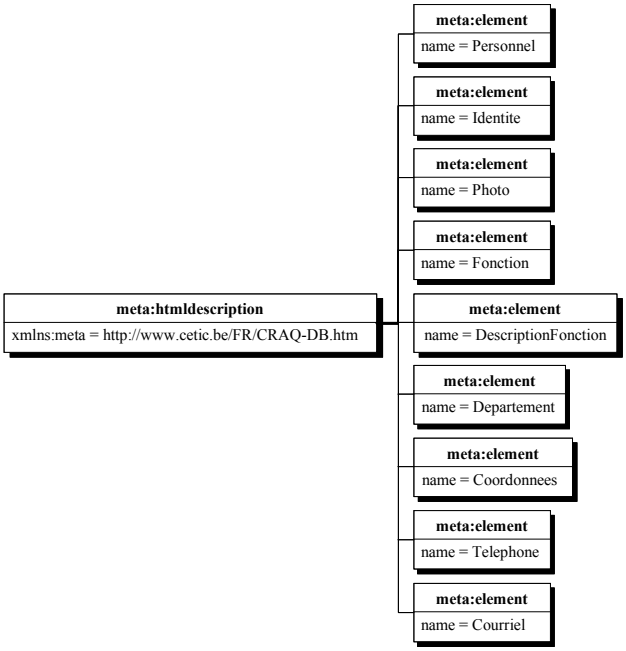


Figure 3.9 - Chaque concept est déclaré comme fils direct de la racine.

La référence à un concept : l'attribut *ref*

Hormis le concept principal, chaque élément déclaré peut faire partie de la définition d'un autre élément (dans notre exemple, les concepts *Telephone* et *Courriel* font partie du concept *Coordonnees*).

Pour représenter ce phénomène (figure 3.10), on insère dans la définition du concept englobant (*Coordonnées*) un <*meta:element/*> pour chaque sous-concept (*Telephone* et *Courriel*). On donne à chacun un attribut *ref* dont la valeur est le nom du concept qu'il référence. Par conséquent, le domaine de valeur de l'attribut *ref* dans un fichier *Meta* est inclus dans le domaine de valeur de l'attribut *name* dans le même fichier.

L'utilisation du mécanisme de référence a pour effet d'importer la totalité de la déclaration d'un concept au sein d'un autre.

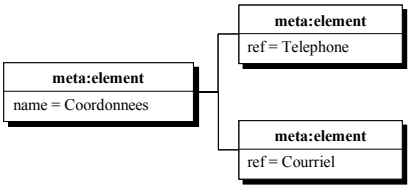


Figure 3.10 - Le concept *Coordonnees* est composé des deux sous-concepts *Telephone* et *Courriel*.

• Les attributs de cardinalité : *min_card* et *max_card*

Les attributs *min_card* et *max_card* sont utilisés pour représenter deux cas particuliers dans le mécanisme de référence, à savoir son aspect facultatif et/ou répétitif.

D'une part, il arrive qu'un sous-élément d'un concept ne soit présent que dans certaines pages du type (sous-élément *optionnel* ou *facultatif*). Dans notre exemple, nous pouvons imaginer que certaines pages ne mentionnent pas de numéro de téléphone. On note cette particularité en donnant la valeur *0* à l'attribut *min_card* lors de la référence au concept *Telephone*.

D'autre part, il arrive qu'un même sous-élément ait plusieurs instances successives au sein d'un concept (sous-élément *multivalué* ou *répétitif*). Dans notre exemple, nous pouvons imaginer que certains membres du personnel ont deux numéros de téléphone. On utilise l'attribut *max_card* pour représenter cette particularité. Sa valeur précise le nombre maximum d'instances successives d'un même sous-concept. Dans le cas où ce nombre est indéterminé, l'attribut prend la valeur spéciale *N*.

La valeur par défaut des deux attributs est *1*. Ainsi, lorsqu'un sous-élément apparaît une et une seule fois, la mention des attributs de cardinalité est inutile. Ajoutons que la valeur de *min_card* doit toujours être inférieure ou égale à celle de *max_card*.

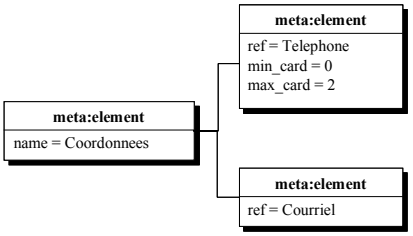


Figure 3.11 - Le sous-élément *Telephone* du concept *Coordonnees* est facultatif et multivalué

• La représentation du balisage *HTML*

Pour rappel, un fichier *Meta* décrit non seulement la sémantique et la hiérarchie des concepts au sein d'un type de pages, mais précise également leur localisation dans la structure *HTML*. A cette fin, le balisage *HTML* des concepts, appauvri, sauf exception, des attributs[5], est repris dans la déclaration des concepts.

Dans notre exemple, les sous-éléments du concept *Coordonnees* sont placés dans un tableau *HTML* (figure 3.12).

5. Les attributs *HTML* étant pour la plupart des éléments de formatage, les conserver dans la description n'apporterait aucune information sémantique ou structurelle. Certains attributs pertinents bénéficieront d'un traitement particulier que nous expliquerons plus bas.

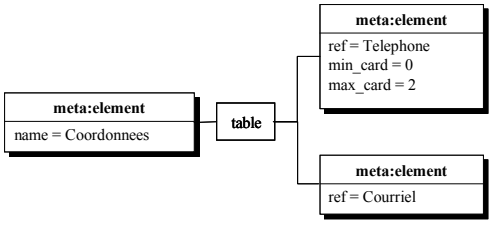


Figure 3.12 - Les concepts *Telephone* et *Courriel* se situent dans un tableau *HTML*.

- **Le repérage des données : <*meta:value*>**

Un élément textuel présent dans une page *HTML* peut être de deux natures. Soit il se répète dans toutes les pages d'un type (constante), soit il est propre à l'instance de page (donnée ou valeur d'instance d'un concept). Cette différence fondamentale doit être prise en compte dans la description *Meta*.

On peut considérer que les constantes sont des éléments de présentation des données au même titre que le balisage. En effet, ils forment ensemble un canevas général dans lequel viennent s'insérer les données. Aussi traitons-nous les constantes textuelles de la même manière que le balisage. Nous les recopions dans le fichier *XML*. (Une illustration commune avec la représentation des données est proposé à la figure 3.13).

Les données, quant à elles, doivent être localisées précisément afin de pouvoir être extraites lors d'une étape ultérieure. Afin de les représenter, nous introduisons un nouvel élément *XML* propre à notre formalisme : l'élément <*meta:value*>.

Etant donné que notre objectif est d'attribuer une valeur sémantique aux informations à extraire, la localisation d'une donnée doit impérativement se faire au sein de la déclaration d'un concept.

Les données à extraire peuvent prendre plusieurs formes :

- un simple élément textuel;
- un mélange de texte et de balises *HTML*;
- une valeur d'attribut.

La donnée est un simple élément textuel

Dans la majorité des cas, les données pertinentes à extraire d'une page sont tout ou partie de simples éléments textuels.

Si la façon dont sont présentées ces données peut fournir des indications sur leur importance dans la page (hiérarchie de concepts notamment), elle n'a par contre aucune influence sur leur valeur intrinsèque. Ainsi dans notre exemple, la donnée «081/990033», associée au concept *Telephone* a la même valeur qu'elle soit affichée en caractères romains, italiques ou gras, qu'elle soit dans une cellule de tableau ou l'élément d'une liste.

Nous choisissons donc d'extraire l'élément textuel sans son balisage extérieur. Dans la description *Meta*, cette valeur est remplacée par l'élément vide <*meta:value*/>.

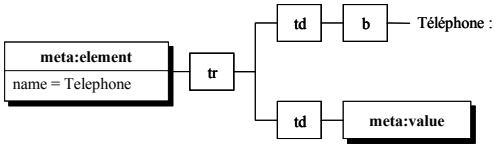


Figure 3.13 - Le concept *Telephone* est placé dans une ligne de tableau. La première cellule contient une constante textuelle («Telephone :») formatée en gras et la seconde une donnée variable.

La donnée est un mélange de texte et de balises *HTML*

Parfois, l'une ou l'autre partie d'une valeur de concept peut se voir attribuer un formatage particulier. Ce sera notamment le cas d'un paragraphe de texte dont certaines zones sont en caractères italiques ou gras. Il convient alors d'utiliser un mécanisme signalant que la donnée est composée de texte et de balises *HTML*.

Pour ce faire, on affecte à l'élément <*meta:value*> un attribut *anyContent* auquel on donne la valeur *true*. [6]

Dans notre exemple, la valeur du concept *DescriptionFonction* est de ce type (figure 3.14).

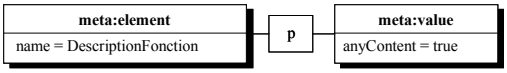


Figure 3.14 - La valeur du concept *DescriptionFonction* est un paragraphe de texte dont certains éléments sont soumis à un formatage particulier.

La donnée est la valeur d'un attribut *HTML*

Comme nous l'avons noté plus haut, la plupart des attributs utilisés dans le balisage *HTML* ne servent qu'à préciser le formatage (couleur de fond, couleur du texte, taille d'une cellule de tableau, etc.).

Cependant, la valeur de certains attributs représente une donnée pertinente à importer dans notre future base de données. C'est le cas notamment de l'attribut *src* de la balise <*img/*> qui renseigne l'adresse où est enregistrée l'image à afficher.

Pour localiser ce type de données nous affectons à l'élément <*meta:value*> un attribut *htmlAttribute* et on lui donne comme valeur le nom de l'attribut qui nous intéresse dans le code source (*src* pour une image par exemple); l'élément *HTML* qui contient l'attribut en question devient le fils de l'élément <*meta:value*>. Dans notre exemple, le concept *Photo* illustre ce cas (figure 3.15).

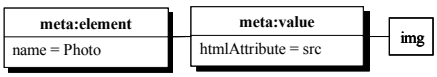


Figure 3.15 - La donnée utile du concept *Photo* est la valeur de l'attribut *src* de la balise <*img/*>.

Notons que ce dernier cas est le seul où l'élément <*meta:value*> peut avoir un fils. En effet, dans les deux autres cas la donnée à extraire est toujours une feuille (ou un bloc cons-

6. La valeur par défaut de cet attribut est *false*; il est donc inutile de le déclarer dans le cas d'une simple donnée textuelle.

tituant une extrémité) de l'arbre syntaxique *HTML*.

Remarques complémentaires concernant les données

Rappelons que le caractère composé et/ou multivalué d'un concept est représenté respectivement par le mécanisme de référence et les attributs de cardinalité. L'élément <*meta:value*> ne peut donc être utilisé dans cette optique. Par conséquent, d'une part, la déclaration d'un concept peut contenir au plus un élément <*meta:value*/>, et, d'autre part un concept ne peut comprendre à la fois un ou plusieurs sous-concepts et un élément <*meta:value*/>.

Notons également que notre formalisme ne permet pas d'isoler une partie seulement d'un élément textuel en tant que donnée à extraire ou de le diviser en plusieurs données. Autrement dit, un élément <*meta:value*> n'aura jamais de frère de type élément textuel.

Nous formulons cette contrainte afin de faciliter l'automatisation du processus d'extraction des données (section 3.6.2). En effet, l'extraction de morceaux d'éléments textuels nécessiterait de préciser au cas par cas ce qu'il faut conserver et ce qu'il faut ignorer. Nous préférons opérer une extraction complète des éléments textuels et appliquer ensuite sur le résultat un processus de nettoyage des données (section 3.6.3).

Afin de prévoir ce post-traitement nous ajoutons aux éléments <*meta:value*/> concernés un attribut *dirty* dont la valeur est *true*.

Citons pour exemple les trois cas les plus fréquents qui nécessitent ce type de traitement :

- la donnée utile n'est qu'une partie de l'élément textuel;
- l'élément textuel contient les valeurs de plusieurs concepts (ex: nom et prénom);
- l'élément textuel contient les différentes valeurs d'un concept multivalué (ex: plusieurs numéros de téléphone).

• Les zones inintéressantes : <*meta:any*/>

Souvent des parties de pages sont vides d'informations relatives au domaine d'application. C'est typiquement le cas d'une bannière publicitaire ou des liens de navigation intra-page.

Afin d'éviter de recopier inutilement dans le fichier *Meta* des portions de code source dénuées d'intérêt, nous introduisons un nouvel élément *XML* dans notre formalisme : <*meta:any*/>. Cet élément se place comme fils unique d'un élément *HTML* et fait office de joker (ou *wild-card*). Il signale que cet élément peut contenir n'importe quoi et que lui et ses descendants peuvent être ignorés. Etant donné que cet élément remplace le contenu d'un élément, il n'a jamais de frères ou de fils.

Dans notre exemple, nous utilisons ce mécanisme pour cacher la bannière en haut de la page (figure 3.16).

Figure 3.16 - <*meta:any/*> remplace le contenu de la ligne de tableau qui renferme la bannière.

• La représentation des liens *hypertext* : <*meta:keyref*>

Dans le chapitre 2, section 2.1.2, nous avons identifié trois types de liens :

- les liens de navigation;
- les liens externes;
- les liens sémantiques.

Un lien de navigation ne contient pas d'informations à extraire. Aussi, nous le cachons dans la représentation *Meta* au moyen d'un élément <*meta:any*/>.

La cible d'un lien externe ne fait pas partie du domaine d'application. Cependant, il est intéressant de conserver l'adresse référencée. Elle sera probablement réutilisée lors de la création dynamique des pages.

Les liens sémantiques matérialisent des relations entre les différents types de pages. Il doivent donc naturellement être représentés.

L'élément *XML* <*meta:keyref*> propre au formalisme *Meta* est destiné à représenter les liens. Il entoure le balisage du lien et signale par sa présence que la valeur de l'attribut *href* du lien fait partie des données à extraire.

Dans notre exemple, le concept *Departement* contient un lien sous forme de donnée textuelle qui renvoie à la page décrivant le département en question (figure 3.17).

Figure 3.17 - Le concept *Departement* contient un lien sémantique.

• La représentation des ancres : <*meta:key*>

Dans le chapitre 2, section 2.1.2, nous avons identifié deux types d'ancres :

- les ancres de navigation;
- les ancres sémantiques.

Parallèlement aux liens, nous ne retenons pas les ancres de navigation.

Le formalisme *Meta* définit l'élément *XML* <*meta:key*> pour représenter les ancres sémantiques. Son fonctionnement est identique à celui de l'élément <*meta:keyref*>, si ce n'est que l'attribut à extraire est *name*.

Dans notre exemple une ancre est insérée avant l'identité de la personne. Elle est la cible d'un lien dans la page du département qui employe cette personne. Nous déclarons un nouveau concept *AncreIdentite* pour la représenter[7] (figure 3.18).

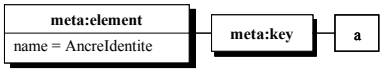


Figure 3.18 - L'élément *AncreIdentite* représente une ancre sémantique

7. Etant donné qu'une ancre est invisible dans un navigateur, nous n'aurions pas pu la repérer en définissant la liste des concepts.

Ajoutons une remarque commune aux éléments <*meta:key*> et <*meta:keyref*>. Etant donné qu'il peuvent tous deux contenir un élément <*meta:value*>, ils ne peuvent en aucun cas cohabiter avec un ou plusieurs <*meta:element*> de référence au sein d'un même concept.

• Les concepts à représentation variable

Rappelons qu'un fichier *Meta* doit décrire toutes les pages d'un même type et non une seule instance. Par conséquent, si la structure ou la présentation d'un concept change d'une page à l'autre, il doit en tenir compte.

Les différences peuvent être de plusieurs natures. Nous en donnons la liste ci-dessous et décrivons le traitement qu'il convient de leur appliquer.

- différence de présentation d'un concept;
- différence d'agencement des sous-éléments d'un concept;
- différence de contenu sémantique d'un concept.

Différence dans la présentation d'un concept

Certains éléments d'un type de pages peuvent être affichés de manière différente d'une page à l'autre. Imaginons dans notre exemple que les constantes de texte dans les concepts *Telephone* et *Courriel* soient formatées tantôt en gras, tantôt en italique. Afin de représenter ces différences, nous ajoutons deux nouveaux éléments *XML* à notre formalisme.

L'élément <*meta:choice*> se positionne comme premier fils dans la déclaration d'un élément. Il signale que le concept a différentes descriptions possibles.

Chacune de ces descriptions prend place dans un élément <*meta:group*> qui obéit aux règles de construction propres à la déclaration des concepts. L'ensemble des éléments <*meta:group*> deviennent les fils de l'élément <*meta:choice*>.

Ajoutons que, par souci de clarté, on évite d'inclure une nouvelle structure de choix dans un élément <*meta:group*>. La figure 3.19 illustre la structure de choix pour le concept *Telephone*.

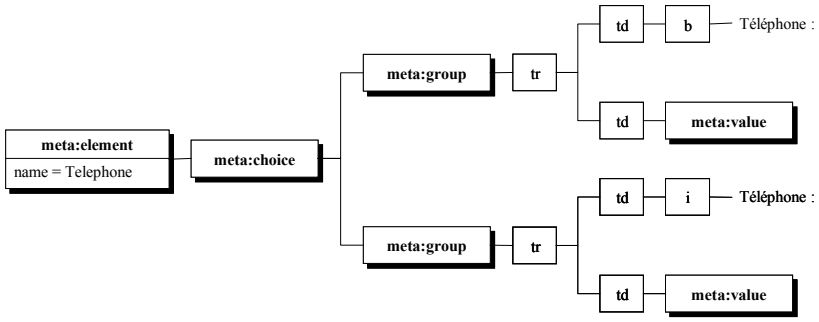


Figure 3.19 - Représentation d'une structure de choix pour le concept *Telephone*

Différence dans l'ordre des sous-éléments d'un concept

Une différence se manifeste également lorsque l'ordre d'apparition des sous-concepts d'un élément varie d'une page à l'autre. Dans notre exemple, on peut imaginer que le con-

cept *Courriel* apparaisse avant le concept *Telephone* dans certaines pages.

C'est à nouveau la structure de choix qui nous permet de représenter ce phénomène.

Différence dans le contenu sémantique d'un concept

Parfois, un même concept présente un contenu sémantique différent d'une page à l'autre. Ce serait le cas dans notre exemple si le concept *Coordonnees* contenait soit un numéro de téléphone, soit un courriel, mais pas les deux.

Pour représenter ce type de cas, nous utilisons à nouveau la structure de choix.

• Remarque sur les différences intra-pages

Dans la section précédente, nous avons présenté les différences qui peuvent survenir entre les pages d'un même type.

Hormis celles-ci, des différences peuvent se manifester également au sein d'une page lorsqu'un concept y apparaît à différents endroits[8] (différence intra-page).

Le cas typique d'une différence intra-page concerne son titre. Souvent, en effet, le titre de la page (marqué par le balisage *HTML* <*title*> au sein de l'élément <*head*>) apparaît ailleurs dans la page.

Afin de préserver la cohérence des données, il est indispensable que la valeur d'instance d'un concept soit identique dans toutes ses occurrences au sein de la page. Cette caractéristique ne pouvant être garantie a-priori dans l'ensemble des pages du type, nous faisons le choix de déclarer un <*meta:element*> différent pour chaque occurrence du concept. La phase de conceptualisation permettra d'éliminer les éventuelles redondances.

A ce stade, nous avons passé en revue l'ensemble des éléments et attributs propres à notre formalisme. Dans la section suivante, nous décrivons la marche à suivre pour l'élaboration d'un fichier *Meta*.

3.5.2 L'élaboration d'un fichier *Meta*

Rappelons une nouvelle fois que chaque instance d'un type de pages doit pouvoir trouver sa description propre dans le fichier *Meta*.

Hormis le cas où toutes les pages d'un type ont une structure et une présentation strictement identiques, l'analyse d'une seule page ne permet pas d'élaborer un *Meta* exhaustif.

A l'opposé, s'il est envisageable d'analyser en parallèle toutes le pages d'un type s'il n'y en a que trois ou quatre, cela devient impensable si leur nombre est important.

Par conséquent, l'élaboration d'un fichier *Meta* se déroule généralement en plusieurs étapes, chacune de celles-ci fournissant un prototype de plus en plus complet.

Lors de la première étape, nous construisons la description d'une page prise au hasard. Ensuite, nous menons une étape de validation. Lors de celle-ci, nous confrontons les autres pages au prototype afin de découvrir si il les décrit correctement.

Lorsque la validation échoue, on modifie le prototype afin d'y intégrer la différence constatée et nous obtenons un nouveau prototype.

---

8. Attention, nous ne parlons pas ici d'un concept multivalué (représenté par les attributs de cardinalité), mais d'un concept dont la même instance se retrouve à plusieurs endroits de la page.

On procède itérativement de la sorte jusqu'à ce que la validation réussisse pour toutes les pages. Le dernier prototype devient alors le fichier *Meta* définitif. La section 3.20 représente l'arbre d'un fichier *Meta* complet pour notre exemple.

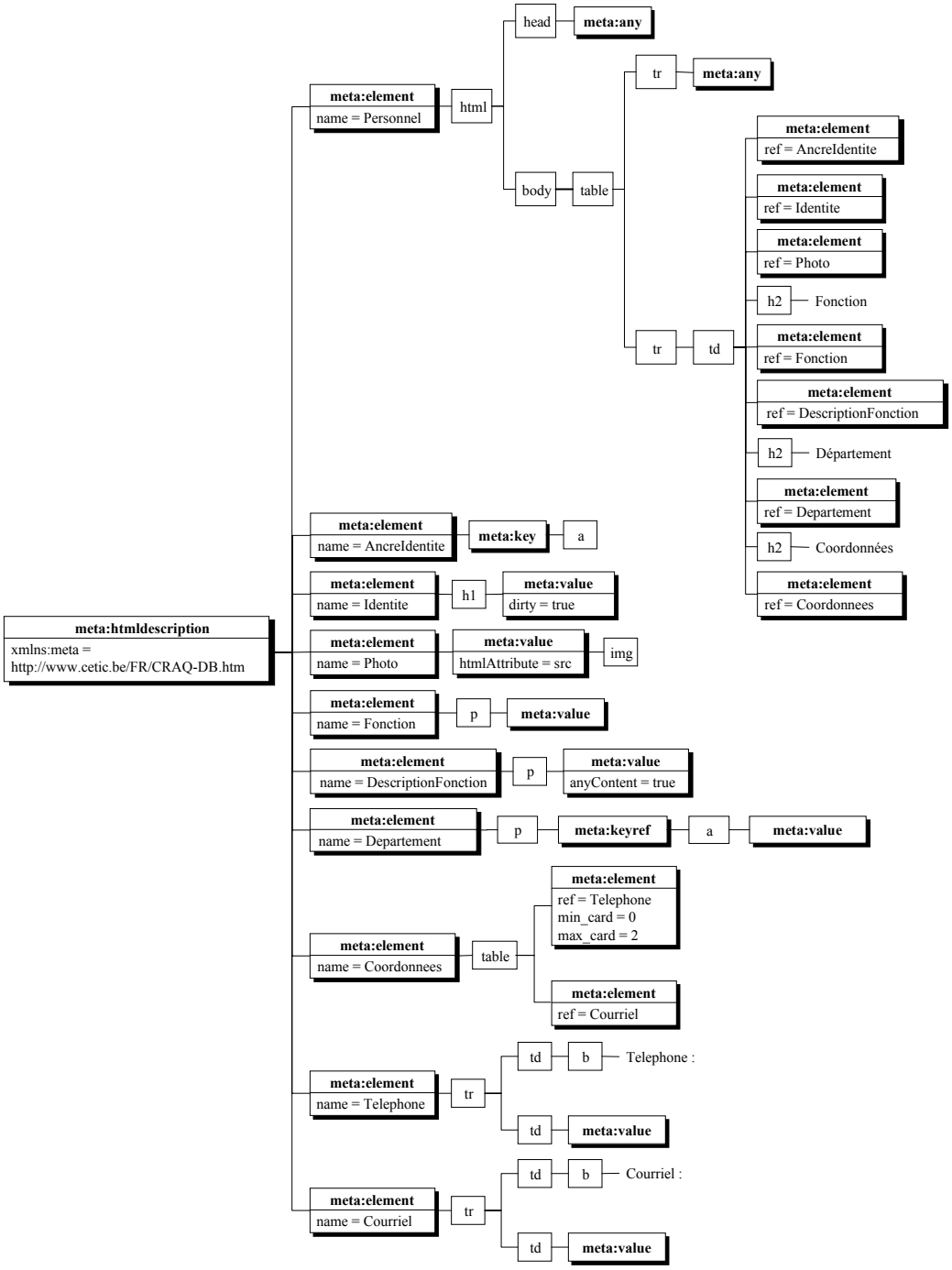


Figure 3.20 - Représentation graphique d'un fichier *Meta* pour le type de pages *Personnel*

3.6 Phase d'extraction

3.6.1 Extraction d'un *XML Schema*

Au départ du fichier *Meta* définitif, nous générons un schéma physique répondant à la norme *XML Schema* qui représente la structure de données du type de pages considéré. Les fichiers *XML* contenant les données extraites des pages (voir section 3.6.2) auront une structure conforme à ce *XML Schema.*

Lors de cette étape, tous les éléments de balisage *HTML* inclus dans la description *Meta* disparaissent. On ne conserve que la structure et les informations sémantiques dégagées au cours de la phase précédente.

La formalisme *Meta* a été développé de façon à faciliter la représentation de la structure des données dans la norme *XML Schema.* Reste à définir des règles de traduction des structures *Meta* en *XML Schema.*

Dans les paragraphes suivants, nous décrivons ces règles. Elles font appel aux notions présentées au chapitre 2, dans la section d'introduction à la norme *XML Schema.* Nous reprenons l'exemple utilisé précédemment pour illustrer nos propos (voir figure 3.5).

Des représentations graphiques des structures crées sont proposées tout au long de cette section[9]. La figure 3.21 donne la signification des éléments utilisés.

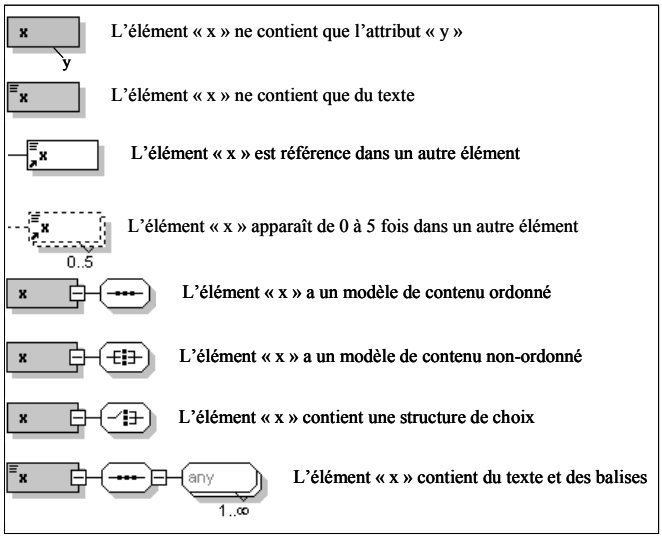


Figure 3.21 - Signification des éléments graphiques utilisés

- **Elément racine du schéma**

Conformément aux spécifications de la norme *XML Schema*, l'élément racine est invariablement <*xsd:schema*>. Il contient une déclaration d'espace de noms qui est utilisé pour qualifier tous les éléments réservés de la norme. Dans notre démarche, chaque concept est

9. Pour les amateurs, le code *XML Schema* correspondant aux graphiques est disponible à l'Annexe A, section A.2 .

déclaré comme fils direct de cette racine.

- **Premier fils de la racine**

Comme dans le fichier *Meta*, nous plaçons comme premier fils de la racine un élément qui décrit le concept principal. Afin d'identifier les différentes pages d'un type, on lui donne un attribut *pageURL* qui représente l'adresse du fichier source. Hormis cet attribut, la description du concept principal suit les règles génériques applicables à tous les concepts (voir ci-dessous).

- **Concept contenant pour seul élément réservé une <*meta:value*>**

La représentation d'un concept dont le seul élément propre au formalisme *Meta* est une <*meta:value*> dépend de la présence ou non d'attributs.

Si l'élément <*meta:value*> n'a pas d'attributs, il correspond à un élément du type prédéfini chaîne de caractères dans la norme *XML Schema*. Dans notre exemple, c'est le cas pour le concept *Fonction* (figure 3.22)

Figure 3.22 - Le concept *Fonction* est du type prédéfini chaîne de caractères.

On applique le même traitement lorsque l'élément <*meta:value*> a un attribut *htmlAttribute*. En effet, dans ce cas la donnée est également une chaîne de caractères (la valeur de l'attribut qu'il faut extraire).

Si l'élément <*meta:value*> a la valeur *true* pour l'attribut *anyContent*, il correspond à un élément de type complexe et de contenu mixte (texte et balises *HTML*). C'est le cas du concept *DescriptionFonction* dans notre exemple (figure 3.23)

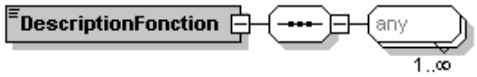


Figure 3.23 - Le concept *DescriptionFonction* est de type complexe et de contenu mixte.

Si l'élément <*meta:value*> a la valeur *true* pour l'attribut *dirty*, il correspond à un élément de type complexe et de contenu simple défini par extension du type prédéfini chaîne de caractères.

- contenu simple : il ne contient pas de sous-éléments
- extension : en plus du texte, il contient un attribut

Dans notre exemple, c'est le cas du concept *Identite* qui contient un nom et un prénom dans un seul élément textuel.

Figure 3.24 - Le concept *Identite* est de type complexe et de contenu simple.

• Concept composé d'un ou plusieurs sous-concepts

Un <*meta:element*> composé d'un ou plusieurs sous-concepts correspond dans la norme *XML Schema* à un élément de type complexe et de contenu complexe ordonné.

- contenu complexe : il contient des sous-éléments
- modèle de contenu ordonné : l'ordre des sous-éléments est imposé

Ce modèle est d'application pour le concept *Coordonnees* de notre exemple (figure 3.25).

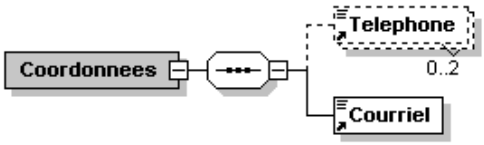


Figure 3.25 - L'élément *Coordonnees* contient deux sous-éléments dont le premier est facultatif et multi-valué.

• Concept contenant une ancre et/ou un lien

Un concept contenant une ancre et/ou un lien aura une représentation différente selon qu'il contienne un élément <*meta:value*> ou pas.

Si il ne contient pas d'élément <*meta:value*>, il correspond dans le modèle *XML Schema* à un élément de type complexe et de contenu vide (composé uniquement d'attributs) des attributs). Dans notre exemple, le concept *AncreIdentite* a cette forme.

Figure 3.26 - Le concept *AncreIdentite* ne contient que l'attribut *key*.

Si, par contre, le concept contient une <*meta:value*>, cela correspond à un élément de type complexe et de contenu simple défini par extension du type prédéfini chaîne de caractères. La représentation est identique à celle de l'attribut *dirty*.

• Concepts incluant un élément <*meta:choice*>

Les éléments comprenant une structure de choix (élément <*meta:choice*>) font l'objet d'un traitement particulier selon le ou les types de différences qu'ils représentent. La figure 3.27 résume les cas possibles et le traitement adéquat à appliquer

présentation	ordre	sémantique	traitement
oui	non	non	1
non	oui	non	2
non	non	oui	3
oui	oui	non	2
oui	non	oui	3
non	oui	oui	3
oui	oui	oui	3

Figure 3.27 - tableau récapitulatif des types de différences.

• Traitement 1

Ce traitement s'applique lorsque les différences concernent uniquement le balisage *HTML*.

Dans ce cas la différence entre les <*meta:group*> n'a aucun intérêt pour le schéma physique, vu que les balises *HTML* n'y sont pas représentées. Aussi, on ne retient que le contenu du premier <*meta:group*> et on lui applique les traitements décrits dans les paragraphes précédents.

• Traitement 2

Ce traitement s'applique lorsque se manifestent des différences dans l'ordre d'apparition des sous-concepts et qu'il n'y a aucune différence sémantique.

Contrairement au premier traitement, les représentations alternatives doivent ici être traduites dans le schéma physique. En effet, les fichiers *XML* contenant les données extraites (voir section 3.6.2) doivent trouver une représentation adéquate dans le *XML Schema*.

Précisons qu'un seul <*meta:group*> suffit à nous fournir tous les renseignements sur le contenu du concept, vu que la différence concerne seulement l'agencement et non la nature des sous-concepts.

Dans la norme *XML Schema*, ce cas correspond à un élément de type complexe et de contenu complexe non-ordonné.

- contenu complexe : il contient des sous-éléments
- modèle de contenu non-ordonné : l'ordre des sous-éléments n'est pas imposé

Ce cas se présenterait dans notre exemple si l'ordre des sous-éléments *Telephone* et *Courriel* du concept *Coordonnees* changeait d'une page à l'autre (figure 3.28).

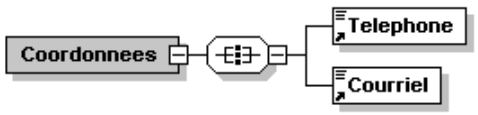


Figure 3.28 - L'ordre des sous-éléments du concept *Coordonnees* change d'une page à l'autre.

Notons toutefois que dans la norme *XML Schema* le modèle de contenu non-ordonné ne peut contenir d'éléments multivalués (si un des sous-éléments est multivalué, le troisième traitement est d'application).

- **Traitement 3**

Ce traitement s'applique lorsqu'une différence d'ordre sémantique survient.

Dans ce cas, les différents <*meta:group*> doivent être traduits dans le schéma physique sous la forme d'une structure de choix propre à la norme *XML Schema*.

Ce cas se présenterait dans notre exemple si le concept *Coordonnees* contenait soit un numéro de téléphone, soit un courriel, mais pas les deux. (figure 3.29).

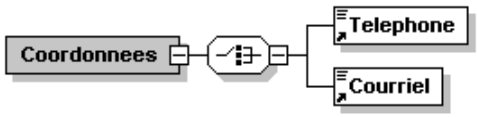


Figure 3.29 - Le concept *Coordonnees* contient une structure de choix.

La figure 3.30 donne la représentation complète du *XML Schema* pour notre exemple.

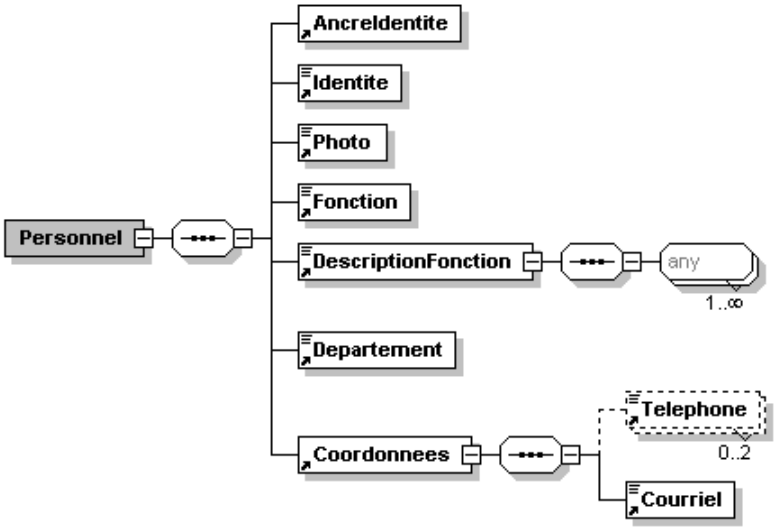


Figure 3.30 - Représentation du *XML Schema* de notre exemple.

3.6.2 Extraction des données

Grâce aux informations structurelles et sémantiques présentes dans le fichier *Meta*, nous extrayons les données de chaque page *HTML* dans un document *XML*. La structure de ce dernier est conforme au *XML Schema* produit dans la section précédente. Les noms des éléments de ce fichier *XML* correspondent aux noms donnés aux concepts dans le fichier *Meta*.

La figure 3.31 montre le code *XML* produit pour la page de la figure 3.5.

```
<?xml version="1.0" encoding="UTF-8"?>
<Personnel pageURL="www.company.com/Personnel/animbus.htm">
    <AncreIdentite keyref="animbus"/>
    <Identite dirty="true">Alfred Nimbus</Identite>
    <Photo>www.company.com/images/animbus.jpg</Photo>
    <Fonction>Responsable informatique</Fonction>
    <DescriptionFonction>
        Le <i>responsable informatique</i> est en charge de la maintenance et du
        renouvellement du matériel informatique du département. Il est en outre
        responsable du bon fonctionnnement et de la sécurité du réseau interne.
    </DescriptionFonction>
    <Departement keyref="www.company.com/Departement/technique.htm">
        Service technique
    </Departement>
    <Coordonnees>
        <Telephone>081/990033</Telephone>
        <Courriel>animbus@company.com</Courriel>
    </Coordonnees>
</Personnel>
```

Figure 3.31 - Fichier*XML* contenant les données de la page «ANimbus.htm».

3.6.3 Nettoyage des données et adaptation du *XML Schema*

Le nettoyage des données consiste à appliquer un traitement spécifique à tous les éléments du fichier *XML* de données qui contiennent un attribut *dirty* dont la valeur est *true*.

Lors de la description du formalisme *Meta*, nous avons signalé trois situations dans lesquelles l'utilisation de l'attribut *dirty* s'imposait :

- la donnée réelle n'est qu'une partie du texte extrait;
- le texte contient les valeurs des sous-concepts d'un concept composé;
- le texte contient les différentes valeurs d'un concept multivalué.

Pour chacun de ces cas, nous allons donner un exemple et décrire le traitement à appliquer.

• La donnée utile n'est qu'une partie du noeud extrait

Le cas typique d'un noeud de texte dont seulement une partie représente la donnée utile est la construction *libellé : valeur*. Nous aurions pu rencontrer ce cas dans notre exemple si la constante et la donnée du concept *Telephone* n'avaient pas été séparées dans deux cellules d'un tableau.

Le traitement à appliquer dans ce cas consiste à ôter de la donnée la sous-chaîne de caractères qui ne nous intéresse pas et à supprimer l'attribut *dirty*.

Afin que les documents de données restent conformes au *XML Schema*, il faut également supprimer la mention de l'attribut *dirty* dans celui-ci.

• La donnée contient les valeurs d'un concept composé

Dans notre exemple, le contenu du concept *Identite* illustre le cas d'une donnée compo-

sée. En effet, la donnée de ce concept contient le nom et le prénom de la personne.

Ce cas nécessite un traitement plus complexe. En effet, l'élément *Identite* est composé de deux sous-éléments (un prénom et un nom). Il faut donc modifier la structure même du fichier *XML* (et celle du *XML Schema*) afin de donner deux fils à l'élément *Identite*.

- **La donnée contient les valeurs d'un concept multivalué**

Ce cas se présente lorsqu'un noeud de texte contient plusieurs valeurs de même nature. Dans notre exemple, nous aurions rencontré ce cas si il y avait eu plusieurs numéros dans la donnée du concept *Telephone*.

Dans ce cas, il faut également modifier la structure du fichier de données (et celle du *XML Schema*) afin d'ajouter un fils multivalué à l'élément *Telephone*.

3.7 Intégration et conceptualisation

Une fois l'ensemble des étapes précédentes réalisé pour tous les types de pages du site internet, il convient d'intégrer les structures dégagées en un schéma unique représentant le domaine d'application.

Il n'est pas suffisant d'intégrer les différents *XML Schemas* en un schéma global unique répondant à la même norme. En effet, il faut laisser le choix aux commanditaires quant à la solution technologique à utiliser pour la création de la future base de données.

Pour ce faire, nous exécutons un travail de conceptualisation qui mène à représenter l'entièreté du domaine d'application dans un schéma indépendant de toute technologie. Par normalisé, nous entendons un schéma qui satisfait à certains critères de qualité tels que la lisibilité, l'expressivité et la minimalité (voir [Hainaut 2001] page 6.40).

Comme nous l'avons signalé au chapitre 2, section 2.2.3, nous nous appuyons sur le modèle *GER* pour réaliser cette étape.

3.7.1 Représentation d'un *XML Schema* dans le modèle *G.E.R*

Nous n'allons pas ici redéfinir l'ensemble des conventions de représentation d'un *XML Schema* dans le modèle *GER*, mais tenons tout de même à présenter par l'exemple les structures utiles pour notre démarche[10]. Le lecteur désireux d'obtenir plus de détails peut consulter [Delcroix] et [François].

Pour chaque élément utilisé, nous donnons une brève description suivie de sa représentation correspondante dans le modèle *GER*.

- **Eléments de type simple**

Un élément de type simple correspond dans un fichier de données à un élément contenant exclusivement une donnée textuelle.

10. En effet, nous n'utilisons dans *RetroWeb* qu'un partie du modèle *XML Schema*.

Dans le schéma *GER*, on représente cet élément par un type d'entités qui porte le nom de l'élément. Son contenu est traduit par un attribut sans nom auquel on associe un stéréotype[11] *«content»*. Celui-ci signale que le contenu est uniquement du texte.

Dans notre exemple, le concept *Fonction* est de ce type (figure 3.32).

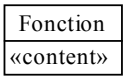


Figure 3.32 - Représentation d'un élément de type simple.

• Eléments de type complexe

Eléments de contenu vide

Un élément de type complexe et de contenu vide correspond dans un fichier de données à un élément qui ne contient que des attributs.

Dans le schéma *GER*, on représente cet élément par un type d'entités qui porte son nom et reprend ses attributs. Par défaut, les attributs sont considérés comme obligatoires. Dans le cas d'un attribut facultatif, on le signale en notant sa cardinalité.

Dans notre exemple, le concept *AncreIdentite* est de ce type (figure 3.33).

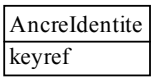


Figure 3.33 - Représentation d'un élément de type complexe et de contenu vide

Eléments de contenu simple

Un élément de type complexe et de contenu simple correspond dans un fichier de données à un élément qui contient du texte et des attributs.

Dans le schéma *GER*, on représente cet élément par un type d'entités qui porte son nom et reprend ses attributs. Nous lui ajoutons un attribut sans nom comme pour les éléments simples afin de représenter la donnée textuelle.

Dans notre exemple, le concept *Departement* est de ce type (figure 3.34).

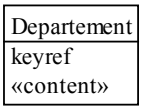


Figure 3.34 - Représentation d'un élément de type complexe et de contenu simple.

Eléments de contenu mixte

Un élément de type complexe et de contenu mixte correspond dans un fichier de don-

11. On utilise un stéréotype pour associer à un objet des caractéristiques non-prédéfinies dans le modèle *GER*

nées issu de notre analyse à un élément dont la donnée est composée de texte et d'éléments *HTML*.

Dans le schéma *GER*, on représente cet élément par un type d'entités qui porte son nom. Son contenu est traduit par un attribut sans nom auquel on associe un stéréotype *«anyContent»*. Celui-ci signale que le contenu est composé de texte et d'éléments *HTML*.

Dans notre exemple, le concept *DescriptionFonction* est de ce type (figure 3.35).

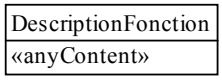


Figure 3.35 - Représentation d'un élément de type complexe et de contenu mixte

Eléments de contenu complexe

Un élément de type complexe et de contenu complexe correspond dans un fichier de données à un élément composé de sous-éléments et éventuellement d'attributs.

Dans le schéma *GER*, on représente l'élément père et chacun des éléments fils par un type d'entités qui porte son nom. Un élément ne pouvant être composé à la fois de sous-éléments et de données, le type d'entités père n'a pas d'attribut.

Pour représenter la relation hiérarchique entre les éléments, on relie chaque fils à son père par un type d'associations binaire qu'on appelle *type d'associations hiérarchique*.

Etant donné que le formalisme *Meta* empêche un élément d'être référencé dans plusieurs autres éléments (i.e. d'avoir plusieurs pères potentiels), la cardinalité du rôle fils dans un type d'associations hiérarchique est toujours [1-1].

La cardinalité minimale et maximale du rôle père correspond aux valeurs des attributs de cardinalité du fils dans le *XML Schema*. On attribue au rôle père le nom *f* (pour *father role*) afin d'éviter la confusion avec le rôle fils dans des schémas complexes.

Reste à représenter le modèle de contenu de l'élément père (ordonné ou non-ordonné). Un modèle de contenu non-ordonné ne nécessite pas de représentation particulière. En effet, dans le modèle *GER*, l'ordre des rôles joués par un type d'entités est indéterminé.

Pour représenter le modèle de contenu ordonné, nous définissons dans le type d'entités père un groupe[12] constitué de l'ensemble des rôles joués par ses fils. On associe à ce groupe une contrainte *seq* qui impose un ordre sur les rôles. Si toutefois le type d'entités père a un seul fils, l'ordre est assuré de fait et il est donc inutile de définir un tel groupe.

Dans notre exemple, le concept *Coordonnées* est de ce type (figure 3.36).

12. Un groupe est un ensemble d'attributs, de rôles ou d'autres groupes qu'on associe à un objet (ici un type d'entités). Il représente un comportement particulier que les membres du groupe doivent respecter par rapport à l'objet auquel le groupe est associé.

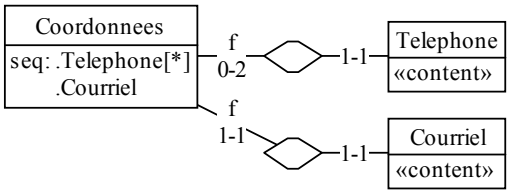


Figure 3.36 - Représentation d'un élément de type complexe et de contenu complexe

Eléments contenant une structure de choix

Un élément contenant une structure de choix correspond à un concept dont les sous-éléments changent d'une page à l'autre.

L'élément père et chacun des fils de la structure de choix sont représentés par un type d'entités qui porte leur nom respectif. Chacun des fils est relié à l'élément père par un type d'associations hiérarchique.

Pour représenter le choix, nous définissons dans le type d'entités père un groupe constitué de l'ensemble des rôles joués par ses fils. On associe à ce groupe une contrainte *choice* qui signifie que le type d'entités est associé à un seul des éléments du groupe. La cardinalité des rôles pères correspond aux attributs de cardinalité de ses fils dans le *XML Schema*.

Pour illustrer ceci, supposons que dans notre exemple, le concept *Coordonnées* contienne soit un ou deux numéros de téléphone, soit une adresse électronique, mais pas les deux. La figure 3.37 illustre ce cas.

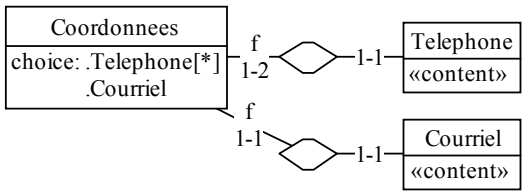


Figure 3.37 - Représentation d'une structure de choix.

Dans le cas où l'une des branches du choix est un élément complexe (i.e. une suite de concepts), elle est représentée par un type d'entités de nom quelconque. On lui associe un stéréotype *«tech»* qui signale que ce type d'entités est artificiellement ajouté pour représenter un ensemble d'éléments.

Pour illustrer ce cas, imaginons que dans certaines pages, les concepts *Telephone* et *Courriel* soient remplacés par une phrase contenant un lien vers une personne de contact. La figure 3.38 illustre ce cas.

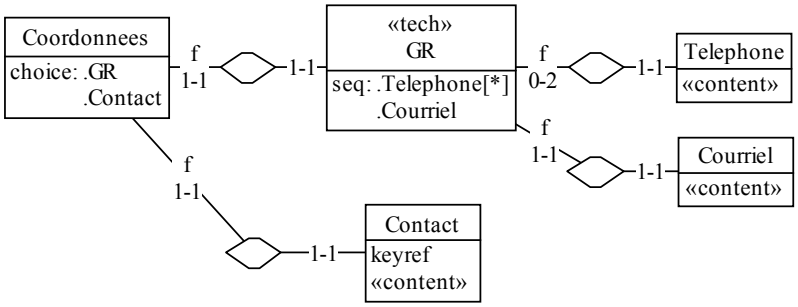


Figure 3.38 - Une structure de choix contenant un élément complexe

3.7.2 Conceptualisation des structures de données

• Remarque préalable

Le schéma physique extrait (i.e. le *XML Schema*) peut être également considéré, dans notre démarche, comme un schéma logique non optimisé de la structure de données. En effet, n'étant pas issu d'un processus d'ingénierie des bases de données mais d'un ensemble de documents bruts, il ne comprend pas d'éléments d'optimisation ni de structures techniques telles que des index ou des espaces de stockage.

• Traduction inverse du schéma logique

Comme le dit J.-L. Hainaut dans [Hainaut 2002] p4-9, *«The logical schema is the technical translation of conceptual constructs»*. Ainsi, au cours du processus de traduction inverse du schéma logique, *«the analyst identifies the traces of such translations, and replaces them with their original conceptual constructs»*.[13]

Dans notre cas, il faut remplacer les constructions propres au modèle *XML Schema* en d'autres constructions indépendantes de ce modèle. Nous devons donc notamment :

- Supprimer la hiérarchie entre les types d'entités;
- Transformer les contraintes propres au modèle *XML Schema* en contraintes du modèle *GER*

Suppression de la hiérarchie

Comme nous l'avons vu dans la section 3.7.1, dans un schéma *GER* conforme au modèle *XML Schema*, chaque type d'éléments est représenté par un type d'entités relié à son père par un type d'associations hiérarchique[14].

Dans un schéma issu de notre méthode, seuls les types d'entités feuilles de la hiérarchie possèdent des attributs[15]. Ceux-ci représentent l'ensemble des informations caractérisant

13. Traduction libre : *«Le schéma logique est une traduction technique de constructions conceptuelles...l'analyste identifie les traces de ces traductions et les remplace par leurs constructions conceptuelles d'origine.»*
14. A l'exception du concept principal qui englobe tous les autres et qui, par conséquent, n'a pas de père.
15. A l'exception du type d'entité représentant le concept principal qui a un attribut contenant l'adresse de la page source.

leur père respectif et, par transitivité, le concept principal du type de pages. D'une manière générale, les types d'entités feuilles peuvent donc être considérés comme des attributs du type de pages. La figure 3.39 donne un petit exemple de hiérarchie.

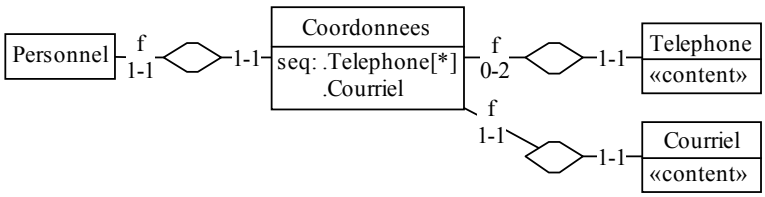


Figure 3.39 - Exemple de hiérarchie de types d'entités

Le type d'entités *Coordonnees* est le père des deux types d'entités *Telephone* et *Courriel*. Ces derniers sont les feuilles de la hiérarchie.

Nous transformons les types d'entités feuilles en attributs de leur père. La figure 3.40 montre le schéma issu de ces transformations.

Le concept *Coordonnées* est devenu une nouvelle feuille de l'arbre hiérarchique.

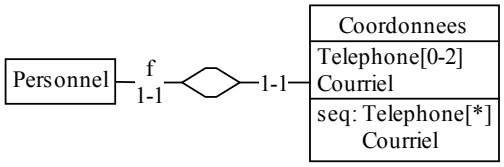


Figure 3.40 - Schéma après les transformations

Nous reproduisons itérativement la même transformation pour les nouvelles feuilles de l'arbre jusqu'à atteindre la racine.

Notons que la transformation appliquée aux fils (transformation d'un type d'entités en attribut d'un type d'entités) est réversible (voir [Hainaut 1995] chapitre 5). Il n'y a donc aucune perte d'informations.

Transformation des contraintes

Dans la section 3.7.1, nous avons traité trois types de contraintes propres au modèle *XML Schema* et avons donné leur représentation dans le modèle *GER* :

- Le modèle de contenu ordonné
- Le modèle de contenu non-ordonné
- La structure de choix

Le modèle de contenu ordonné (groupe seq)

Représenté par la contrainte *seq*, le modèle de contenu ordonné impose l'ordre d'apparition des fils d'un type d'entités père. La transformation de cette contrainte est automatiquement résolue lors de la suppression de la hiérarchie. En effet, en transformant les fils en attributs du type d'entités père selon l'ordre stipulé par le groupe *seq*, ils

apparaissent dans cet ordre au sein de leur père (voir figure 3.40). Le groupe *seq* peut donc être supprimé.

Le modèle de contenu non-ordonné

Le modèle de contenu non-ordonné ne nécessitait pas une représentation particulière dans le schéma *GER*. En effet, il n'y a pas d'ordre imposé sur les rôles.

La transformation des types d'entités en attributs de leur père introduit la notion d'ordre. Par conséquent, le modèle non-ordonné n'est plus représenté.

Notons toutefois que ce modèle de contenu a peu d'utilité dans l'optique d'une création dynamique de pages internet. En effet, la génération automatique de la présentation d'un type d'information suivra un modèle et, par conséquent, les différentes informations apparaîtront toujours dans le même ordre.

La structure de choix (groupe choice)

Pour transformer un groupe *choice*, on commence par supprimer la hiérarchie entre le type d'entités qui contient le groupe et ses fils. Ensuite, la modification à apporter dépend de la cardinalité minimum des attributs qui composent le groupe.

Si tous les éléments sont obligatoires, la sémantique du groupe *choice* correspond en tous points à la contrainte d'existence *exactly one* du modèle *GER*. Celle-ci signifie que le type d'entités contient un et un seul des attributs du groupe

Cette contrainte se représente en rendant facultatifs tous les attributs composant le groupe et en modifiant l'intitulé de la contrainte *choice* en *exact-1*.

Dans notre exemple, supposons que les coordonnées contiennent soit un numéro de téléphone, soit un courriel mais pas les deux. La figure 3.41 illustre la transformation à appliquer.

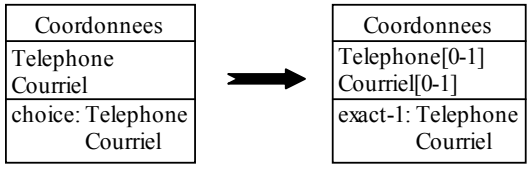


Figure 3.41 - Transformation d'un groupe *choice* en groupe *exact-1*

Si, par contre, au moins un des membres du groupe *choice* est facultatif, cela correspond à la contrainte d'exclusion du modèle *GER*. Celle-ci signifie que le type d'entités contient au plus un des attributs du groupe.

Ce cas se présenterait dans notre exemple si les coordonnées contenaient un numéro de téléphone ou un courriel ou rien du tout.

Après transformation, le schéma est identique à la figure 3.41, si ce n'est que le nouvel intitulé du groupe est *excl* au lieu de *exact-1*.

La figure 3.42 représente le type de pages *Personnel* après la suppression de la hiérarchie et des contraintes propres au modèle *XML Schema*.

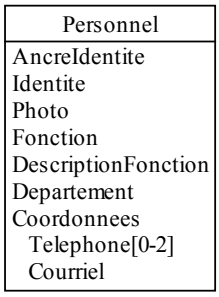


Figure 3.42 - Représentation complète du type d'entités *Personnel*

• Intégration des schémas

Dans la plupart des cas, un site internet présente plusieurs types de pages. Dès lors, afin d'obtenir un schéma unique représentatif du domaine d'application complet, il est nécessaire d'intégrer les différents schémas entre eux.

Pour réaliser ce travail, nous analysons le schéma obtenu à l'étape précédente et vérifions le bien-fondé de nos constatations en consultant des exemples de fichiers de données.

Ce processus pourrait faire l'objet d'un travail exhaustif, aussi allons-nous simplement énoncer quelques principes généraux d'application dans le cadre des analyses de sites internet.

Les types d'informations comparables

Souvent, un site contient plusieurs types d'informations comparables, comme différents types de produits ou différentes classes d'employés.

Dans ce cas, on regroupe dans un sur-type les attributs communs. Chaque sous-type conserve les attributs qui lui sont propres.

Les liens

A ce stade, notre analyse a conduit à la création d'un schéma par type de pages sans représentation des relations entre ces types.

Pour déceler ces relations, on s'appuie sur les informations véhiculées par les liens hypertext à nature sémantique. En effet, un tel lien récurrent dans les pages d'un type référence toujours les instances d'un même type de pages.

L'adresse cible d'un lien est toujours un identifiant de la page référencée (voir chapitre 2, section 2.1.2). Par conséquent, nous pouvons définir une contrainte référentielle[16] entre l'attribut qui représente le lien et celui qui représente la cible.

Les informations redondantes.

Il est fréquent que des informations se répètent dans différents types de pages. (Par

16. Les valeurs prises par l'attribut qui représente le lien sont incluses dans l'ensemble des valeurs prises par l'attribut qui représente la cible.

exemple, une page du type *Departement* cite ses employés, ceux-ci ont une page personnelle). Ce phénomène se rencontre le plus souvent dans les pages mises en relation par des liens. Ces redondances doivent être supprimées dans le schéma final.

La suppression ne peut se faire de manière aléatoire. L'information doit être conservée à l'endroit qui lui sied le mieux. (Par exemple, les informations relatives aux employés doivent être conservées dans le type d'entités *Personnel* plutôt que dans *Departement*).

- **Normalisation conceptuelle**

Le processus de normalisation conceptuelle consiste à donner une apparence définitive au schéma conceptuel brut obtenu lors des étapes précédentes. Il est fortement dépendant du cas analysé et de la sensibilité de l'analyste.

La normalisation conceptuelle pourrait faire à elle seule l'objet d'un travail exhaustif, aussi allons-nous nous limiter à en donner une définition générale empruntée à J.-L Hainaut dans [Hainaut 2002] p 4-9 : *«This process restructures the basic conceptual schema in order to give it the desired qualities one expects from any final conceptual schema, such as expressiveness, simplicity, minimality, readability, genericity, extensibility. For instance, some entity types are replaced with relationship types or with attributes, is-a hierarchies are made explicit, names are standardized, etc.»*[17]

Le lecteur soucieux d'obtenir une description détaillée de ce processus pourra consulter [Hainaut 2002] p. 6-11 à 6-14 et [Hainaut 2000] p. 6-40 à 6-55.

3.8 Conception de la base de données

L'ensemble des étapes qui constituent le processus de création d'une base de données est entièrement orienté par la technologie[18] choisie. Nous allons ici nous limiter à définir de manière générale les étapes successives. Le lecteur désireux d'approfondir le sujet peut consulter [Hainaut 2001] chapitres 8 à 11.

3.8.1 La conception logique

Cette étape consiste à transformer le schéma conceptuel en un schéma équivalent conforme au modèle du *SGBD* choisi.

Par équivalent, nous signifions qu'il doit être construit au moyen de transformations à sémantique constante (réversibles).

Par conforme, nous entendons que l'ensemble de ses structures doit correspondre au modèle du *SGBD* cible (exemple: un schéma respectant les principes énoncés à la section 3.7.1 est conforme au modèle *XML Schema*).

---

17. Traduction libre : *«Ce processus consiste à restructurer le schéma conceptuel brut afin qu'il réponde aux critères de qualité attendus d'un schéma conceptuel final, à savoir l'expressivité, la simplicité, la minimalité, la lisibilité, la généricité et l'extensibilité. Par exemple, on remplace certains types d'entités par des types d'associations ou par des attributs, on rend explicites les relations* is-a*, on normalise les noms des objets, etc.».*
18. En l'occurrence le système de gestion de base de données (*SGBD*) choisi.

3.8.2 La conception physique

La conception physique consiste à augmenter le schéma logique de paramètres et caractéristiques techniques tels que des index, des espaces et des modes de stockage.

Comme le schéma logique, le schéma physique doit être équivalent à ses prédécesseurs et conforme au modèle du *SGBD* cible.

L'équivalence est garantie par le fait qu'il s'agit d'un enrichissement du schéma logique. La suppression des paramètres et caractéristiques techniques du schéma physique restitue le schéma logique original.

3.8.3 Le codage du schéma

Le codage du schéma consiste à traduire le schéma physique dans le langage de description de données (*DDL*) propre au *SGBD* cible afin de créer une structure physique prête à accueillir les données.

3.9 Migration des données

Cette étape consiste à enregistrer les données contenues dans les fichiers *XML* extraits des sources *HTML* dans la structure physique créée à l'étape précédente.

Ce travail peut se faire de manière automatique en définissant des règles de traduction (ou mapping) de la structure *XML* vers la structure de la base de données.

La définition de ces règles se base sur l'historique des transformations appliquées au schéma d'origine (*XML Schema*) lors des phases de conceptualisation, intégration et conception. Grâce à cet historique, on peut retrouver les correspondances entre les structures de départ et d'arrivée et ainsi localiser l'endroit adéquat où enregistrer chaque donnée.

Chapitre 4

Outils de support pour la méthode

Dans ce chapitre, nous décrivons les outils de support pour notre méthodologie. Par souci de cohérence ils sont présentés selon l'ordre de leur utilisation dans le processus. Pour chacun d'entre eux, nous définissons ses objectifs, son fonctionnement et ses éventuelles limites :

4.1 Nettoyage du code *HTML*

4.1.1 TagSoup

- **Objectifs**

Le nettoyage du code *HTML* consiste à transformer les sources du site analysé en documents *XML* bien formés.

Pour mener à bien ce pré-traitement, nous utilisons l'outil libre *TagSoup* [Cowan]. Il s'agit d'un *parser* de code *HTML* écrit en *Java*.

- **Fonctionnement**

TagSoup s'occupe à la fois de corriger des erreurs de syntaxe (guillemets manquants dans les valeurs d'attributs, chevauchement d'éléments, absence de balises de fermeture obligatoires) et de fermer tous les éléments légitimement laissées ouverts dans les fichiers *HTML*. De cette manière, il transforme la source en document *XML* bien formé.

Par défaut, *TagSoup* réserve un traitement particulier aux balises inconnues du langage *HTML*. Il les conserve dans le code, mais les transforme en éléments vides, soit en ramenant la balise de fermeture à côté de celle d'ouverture, soit en ajoutant une telle balise si elle n'existe pas.

Notons toutefois que l'outil permet de contourner cette règle. Il suffit à l'utilisateur de spécifier les jeux de balises qui doivent être traitées comme s'il s'agissait d'éléments de la norme.

- **Limites**

L'outil éprouve parfois des difficultés lorsqu'une valeur d'attribut est précédée de guillemets ouvrants mais dépourvue de guillemets fermants.

4.2 Enrichissement sémantique

4.2.1 Retrozilla

- **Objectif**

Assister l'analyste dans l'élaboration d'une première version d'un fichier *Meta* sur la base d'une instance de page d'un type.

Ce travail est réalisé à l'aide de l'outil *Retrozilla* développé par le *CETIC*. Il s'agit d'un plug-in ajouté à l'éditeur *HTML* intégré au navigateur *Mozilla*.

- **Fonctionnement**

Retrozilla annote le code *HTML* d'informations structurelles et sémantiques propres au formalisme *Meta*.

La figure 4.1 présente le menu proposé par *RetroZilla*. Nous en décrivons les options ci-dessous.

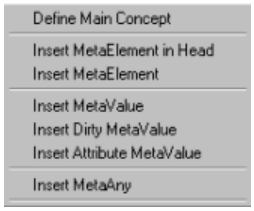


Figure 4.1 - Le menu de *RetroZilla*

Define Main Concept :

L'option *Define Main Concept* permet de définir le concept principal du type de pages. L'utilisateur renseigne le nom choisi dans la boite de dialogue de la figure 4.2 et l'outil ajoute un élément <*meta*> dans l'élément <*head*> du code *HTML* comme suit :

```
<meta name="pagetype" content="NomDuConceptPrincipal">
```

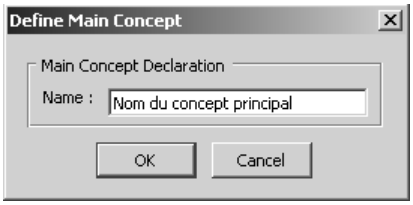


Figure 4.2 - La boite de dialogue *Insert MetaElement*

Notons que la déclaration d'un concept principal est obligatoire. C'est ce concept qui représente le type de pages analysé. Evidemment, il est interdit de définir deux concepts de ce type; aussi, après la définition du concept principal, l'option du menu est désactivée.

Insert MetaElement in Head :

L'option *Insert Meta Element in Head* permet de définir un concept au sein de l'élément *HTML* <*head*> qui est la partie non visible dans une fenêtre de navigateur. L'utilisateur donne le nom du concept dans la boite de dialogue (figure 4.3), sélectionne l'élément *HTML* (<*title*> ou <*meta*>) auquel le concept doit être associé et spécifie éventuellement l'attribut à marquer comme valeur à extraire (dans le cas d'un élément <*meta*>).

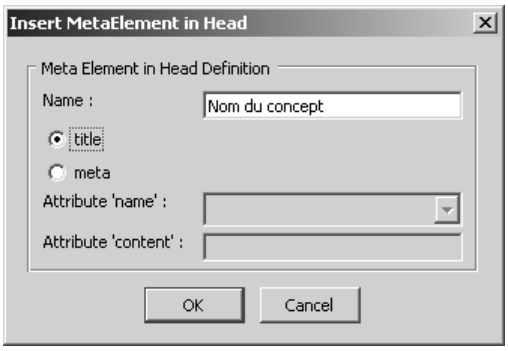


Figure 4.3 - La boîte de dialogue *Insert MetaElement in Head*

L'outil se charge d'englober l'élément *HTML* dans un balisage <*meta:element*> et de marquer la donnée à extraire au moyen d'un balisage <*meta:value*>. Voici un exemple de résultat pour <*title*> :

```
<meta:element name="TitrePage">
    <title>
        <meta:value>Alfred Nimbus</meta:value>
    </title>
</meta:element>
```

Insert MetaElement :

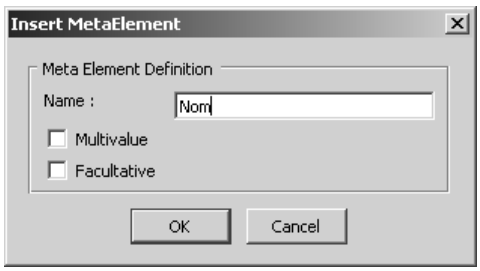


Figure 4.4 - La boîte de dialogue *Insert MetaElemen*t

L'option Insert *MetaElement* permet de définir les concepts visuellement dans la fenêtre de l'éditeur. L'utilisateur sélectionne la zone couverte par le concept, appelle le

menu, donne le nom à affecter au concept dans la boite de dialogue (figure 4.4) et précise au besoin son caractère multivalué et/ou facultatif.

L'outil englobe la zone sélectionnée par un balisage <*meta:element*>. Voici un exemple de résultat après sélection d'un élément *HTML* <*h1*>.

```
<meta:element name="Telephone">
    <h1>081/990033</h1>
</meta:element>
```

Insert MetaValue et Insert Dirty MetaValue:

L'option *Insert MetaValue* permet de localiser une donnée à extraire au sein d'un concept précédemment défini. L'utilisateur sélectionne le noeud de texte (ou l'ensemble des noeuds dans le cas d'une donnée de contenu mixte) et appelle le menu. L'outil se charge d'entourer la sélection d'un balisage <*meta:value*>. Voici le résultat pour l'exemple précédent :

```
<meta:element name="Telephone">
    <h1>
        <meta:value>081/990033</meta:value>
    </h1>
</meta:element>
```

L'option Insert *Dirty MetaValue* fonctionne de la même manière mais permet de signaler en plus que cette donnée doit subir un traitement ultérieur. Pour ce faire, la balise <*meta:value*> reçoit un attribut *dirty* dont la valeur est *true.*

Insert Attribute MetaValue :

L'option *Insert Attribute MetaValue* permet de préciser que la donnée associée à un concept est la valeur d'un attribut. L'utilisateur sélectionne au sein du concept déclaré l'élément *HTML* contenant l'attribut dont il faut extraire la valeur, puis il appelle le menu. Dans la boite de dialogue de la figure 4.5, il choisit dans la liste déroulante l'attribut adéquat.

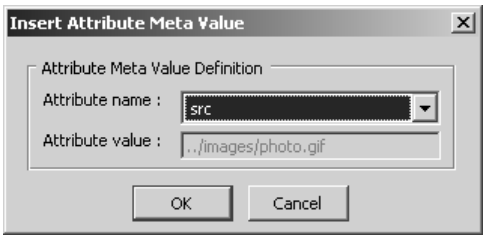


Figure 4.5 - La boite de dialogue *Insert Attribute MetaValue*

L'outil se charge d'entourer l'élément d'un balisage <*meta:value*> précisant le nom de l'attribut.. Voici un exemple pour l'extraction de l'adresse source d'une image :

```
<meta:element name="Image">
    <meta:value htmlAttribute="src"><img src="../images/photo.gif"></meta:value>
</meta:element>
```

Insert MetaAny :

L'option Insert *MetaAny* permet de localiser une zone dénuée d'intérêt pour le processus. L'utilisateur sélectionne l'élément *HTML* dont le contenu doit être ignoré et l'outil se charge d'insérer un balisage <*meta:any*> autour du contenu de ce noeud.

Voici un exemple où l'outil déclare insignifiante une bannière publicitaire placée dans un tableau :

```
<table>
    <meta:any>
        <tr>
            <td><img src="../images/banner.jpg"></td>
        </tr>
    </meta:any>
</table>
```

• Limites

Même s'il fait preuve d'un comportement satisfaisant et d'une grande facilité d'utilisation, *Retrozilla* montre certaines lacunes à l'heure actuelle :

Le traitement d'une page doit se faire en une seule session. En effet, à l'ouverture d'un document, l'éditeur de *Mozilla* l'analyse et le modifie éventuellement afin de le rendre conforme (ou presque) à la norme *HTML 4.01*. Par conséquent, il élimine ou transforme en éléments vides les balises inconnues, et donc toutes celles issues du traitement par *Retrozilla*.

L'impossibilité de réouvrir le fichier annoté empêche également de le corriger avec l'outil lors de la découverte de nouvelles informations obtenues en validant les autres pages du type (nouveau concept, caractère facultatif d'un concept, etc.). Par conséquent, les corrections doivent actuellement être réalisées à la main.

La définition du caractère multivalué ne donne pas toujours les résultats attendus. En effet, l'outil applique le modèle à tous les frères suivants de même nom. Aussi, si le fonctionnement est correct pour les éléments d'une liste ou les lignes d'un tableau, des structures moins ordonnées mènent à des résultats inadéquats. Supposons par exemple un titre suivi d'une série de paragraphes représentant les multiples instances d'un concept de même nature, suivis à leur tour d'un autre titre et de paragraphes représentant un autre concept. La déclaration du premier paragraphe en tant que concept multivalué va reproduire le modèle aux autres paragraphes présents sous le même titre, mais également à ceux qui suivent le titre suivant.

La résolution de ces problèmes est en cours. Si la gestion des concepts multivalués a de bonnes chances d'être rapidement et assez aisément réglée en adaptant le plug-in, pour le reste, il est nécessaire d'apporter des modifications au code source de *Mozilla*, ce qui n'est pas sans risque.

4.2.2 MetaGenerator

• Objectif

Générer automatiquement un prototype *Meta* à partir d'un fichier *HTML* enrichi par l'analyse réalisée avec l'outil *Retrozilla*.

Cette tâche est réalisée par un objet *Java* appelé *MetaGenerator* (développé par le *CETIC*).

• Fonctionnement

L'outil *MetaGenerator* consiste à appliquer une feuille de transformations *XSL* au fichier *HTML* annoté par *Retrozilla* pour en faire un fichier *Meta*. Cependant, comme nous l'avons signalé plus haut, l'éditeur *HTML* intégré à *Mozilla* transforme la source en document conforme ou presque à la norme *HTML 4.01*. Par conséquent, le fichier issu de *Retrozilla* n'est pas bien formé au sens *XML* (par exemple, les balises de fermeture d'éléments tels que <*link*> ou <*meta*> dans l'entête du document *HTML* sont supprimées).

Pour ce faire, l'objet *MetaGenerator* utilise l'outil *TagSoup* décrit à la section 4.1.1 avant d'appliquer la feuille *XSL*. La flexibilité de ce dernier nous permet de lui donner une liste de balises inexistantes dans le langage *HTML* qu'il doit considérer comme valides. Ainsi, les balises ajoutées par *Retrozilla* ne sont pas traitées en tant qu'élément inconnus.

4.2.3 HTMLValidator

• Objectif

Confronter les instances du type de pages analysé par rapport au prototype du fichier *Meta* en vue de compléter ou corriger celui-ci si des différences se manifestent.

Cette tâche est réalisée par un objet *Java* appelé *HTMLValidator* (développé par le *CETIC* et le *LIBD*).

• Fonctionnement

L'outil *HTMLValidator* parcourt en parallèle le concept principal (i.e. le premier fils de la racine) du fichier *Meta* et une des pages données en entrée. Lorsqu'il rencontre un <*meta:element*> de reference, il importe sa définition et continue ainsi de manière récursive. Le parcours est de type *en profondeur d'abord*.

Tout le long de l'analyse, il fournit dans la console de travail des informations sur les éléments analysés. Celles-ci permettront de localiser l'inconsistance en cas d'échec.

Dès qu'il détecte une erreur, l'outil mentionne dans la console l'invalidité de l'élément fautif et de tous ses ancêtres[1], imprime un message de non conformité du fichier, puis passe au fichier suivant. Lorsqu'il a terminé le parcours de toutes les pages, il affiche le nom des pages qui n'ont pu être validées.

1. En effet, l'invalidité d'un élément se répercute automatiquement sur tous ses ancêtres.

4.3 Extraction

4.3.1 SchemaExtractor

- **Objectif**

Générer automatiquement le *XML Schema* correspondant à la version finale d'un fichier *Meta* conformément aux règles de traduction définies au chapitre 3, section 3.6.1.

Cette tâche est réalisée par un objet *Java* appelé *SchemaExtractor* (développé par le *LIBD*).

- **Fonctionnement**

L'outil *SchemaExtractor* génère le *XML Schema* en plusieurs étapes :

1. Il applique une feuille de transformations *XSL* au fichier *Meta* qui lui ôte le balisage *HTML*. Le résultat est enregistré dans un fichier temporaire nommé *Cleaned.xml*.
2. Il applique au fichier temporaire une nouvelle feuille de transformations *XSL* qui génère un premier *XML Schema* correspondant au *Meta*. Il enregistre le résultat dans un fichier appelé *rawSchema.xsd*.
3. Il appelle un objet *Java* appelé *SchemaOptimizer* qui se charge de simplifier certains éléments <*meta:choice*> (élimination des <*meta:group*> redondants, transformation d'une structure <*xsd:choice*> en structure <*xsd:all*> lorsque les différences ne concernent que l'ordre des sous-éléments).

4.3.2 DataExtractor

- **Objectif**

Extraire automatiquement les données depuis les fichiers *HTML* et les structurer conformément au *Meta* dans un fichier *XML* bien-formé et conforme au *XML Schema* généré à l'étape précédente.

Cette tâche est réalisée par un objet *Java* appelé *DataExtractor* (développé par le *CETIC* et le *LIBD*).

- **Fonctionnement**

L'outil *DataExtractor* construit un arbre *XML* qui sera exporté dans un fichier une fois la tâche accomplie.

Avant de parcourir une page, l'outil crée dans l'arbre un élément *XML* dont le nom est celui du type de pages. Cet élément accueillera les données de la page structurées conformément au *XML Schema*.

Ensuite, l'outil parcourt en parallèle le concept principal du fichier *Meta* et une page *HTML*. Chaque fois qu'il rencontre un <*meta:element*> dans le fichier *Meta*, il crée un élément *XML* portant le nom du <*meta:element*>[2]. Lorsqu'il rencontre un élément <*meta:value*> il extrait l'élément correspondant dans le fichier *HTML* et l'insère dans l'arbre *XML*.

Lorsque toute la page a été parcourue, l'arbre *XML* créé est placé comme fils d'une racine artificielle. Celle-ci permet d'extraire les données de toutes les pages du type dans un seul fichier.

L'outil répète ce processus pour toutes les pages du type.

4.3.3 DataCleaner et SchemaAdaptor

• Objectif

Nettoyer et/ou restructurer les données marquées d'un attribut *dirty* dans un fichier *XML* de données et adapter le *XML Schema* en conséquence.

Cette tâche est réalisée par deux objets *Java*, *DataCleaner* et *SchemaAdaptor* (développés par le *LIBD*).

• Fonctionnement

L'outil *DataCleaner* demande trois arguments fixes (type de traitement, type de pages, nom de l'élément à traiter) plus d'autres arguments en fonction du traitement à appliquer (voir chapitre 3, section 3.6.3) :

- lorsque la donnée utile n'est qu'une partie de la donnée extraite, il faut préciser les bornes de la sous-chaîne à conserver;
- lorsqu'il faut diviser une donnée de nature composée ou multivaluée, il faut préciser le type de séparateur entre les éléments.

Lors de l'extraction d'une sous-chaîne, l'outil *DataCleaner* supprime du contenu de l'élément les parties de la chaîne de caractères qui se situent en dehors de bornes précisées dans les arguments. Il enlève également l'attribut *dirty* de l'élément traité.

Lors de la division d'une donnée de nature composée, l'outil crée autant de fils à l'élément traité qu'il y a de sous-éléments séparés par le séparateur donné en argument. Le nom des nouveaux éléments est composé du nom de l'élément de départ suivi de «Child_» et d'un numéro de séquence. Evidemment, il supprime également l'attribut *dirty* de l'élément de départ.

exemple avant traitement

```
<Identite dirty="true">Alfred Nimbus</Identite>
```

exemple après traitement

```
<Identite>
    <IdentiteChild_1>Alfred</IdentiteChild_1>
    <IdentiteChild_2>Nimbus</IdentiteChild_2>
</Identite>
```

Lors de la division d'une donnée multivaluée, le traitement est identique à l'exemple précédent, si ce n'est que chacun des nouveaux éléments a le même nom. Celui-ci est com-

2. Notons deux cas particuliers : 1) si le <*meta:element*> est facultatif et qu'il n'est pas présent dans la page traitée, aucun élément n'est ajouté à l'arbre *XML*. 2) si le <*meta:element*> est multivalué, il crée autant d'éléments qu'il rencontre d'instances dans la page.

posé du nom de l'élément de départ suivi de «Instance».

exemple avant traitement

```
<Telephone dirty="true">081/336699, 081/996633</Telephone>
```

exemple après traitement

```
<TelephoneInstance>
    <TelephoneInstance>081/336699</TelephoneInstance>
    <TelephoneInstance>081/996633</TelephoneInstance>
</TelephoneInstance>
```

Une fois le traitement appliqué au fichier de données, l'outil passe le relai au *SchemaAdaptor* pour qu'il adapte de manière adéquate le *XML Schema* associé (suppression de l'attribut *dirty* et ajout éventuel de sous-éléments à l'élément traité).

• Limites

Actuellement, ces deux outils ne traitent que des cas relativement simples. Ils devront être augmentés d'autres traitements au gré de la rencontre de nouveaux cas.

4.4 DB-Main

• Objectif

DB-Main est un outil *CASE* développé par l' équipe *DB-Main* du *LIBD*. Il s'adresse à l'ingénierie et la rétro-ingénierie des applications de bases de données. Le lecteur intéressé pourra trouver une présentation générale de l'outil sur le site qui y est consacré [DB-main].

Dans le cadre de notre méthodologie, nous allons l'utiliser pour réaliser diverses tâches :

- représenter les structures de données extraites (i.e. les différents *XML Schemas*) dans le modèle *GER;*
- conceptualiser et intégrer ces mêmes structures;
- concevoir un schéma de base de données à partir du schéma conceptualisé et intégré;
- générer le code de cette base de données;
- générer un migrateur pour importer les données du format *XML* vers la base de données construite.

• Limite pour notre utilisation

Actuellement, *DB-Main* dispose d'un outil d'importation pour les *DTDs XML*, mais pas pour les *XML Schemas*. Nous devons donc soit construire à la main le schéma *GER* conforme au modèle *XML Schema*, soit passer par une transformation des *XML Schemas* en *DTDs* avant d'effectuer l'importation.

La transformation en *DTD* provoque des pertes d'informations, notamment :

- dans le modèle *DTD*, les valeurs des cardinalités sont limitées à *0*, *1* et *N*.
- les structures <*xsd:all*> et <*xsd:any*> n'ont pas d'équivalent dans le modèle *DTD*.

Si on choisit cette méthode, il faut s'armer de vigilance afin d'apporter au schéma *GER*

les adaptations adéquates.

Chapitre 5

Etude de cas : *LittleZon*

Dans ce chapitre, nous appliquons notre méthodologie sur un exemple concret : le site *LittleZon*[1]. Ce dernier est un site factice conçu uniquement pour servir d'étude de cas.

Dans un premier temps, nous présentons la situation. Ensuite, nous exécutons chacune des étapes de la méthodologie en nous appuyant sur les outils présentés au chapitre 4. Pour terminer, nous donnons un exemple de page générée à partir de la base de données construite.

5.1 Mise en situation

LittleZon est un petit espace de vente par correspondance de livres, DVDs et disques compacts. L'accès à ses produits se fait exclusivement par son site internet (composé uniquement de pages statiques), les commandes, quant à elles, se réalisent par courriel.

Nouvelle sur le marché, la société bénéficie d'un grand intérêt de la part du public, grâce à un service rapide, des informations pertinentes et un début d'interactivité avec ses clients.

Rassurés par ce succès naissant, les propriétaires souhaitent aujourd'hui se donner davantage de moyens pour grandir, améliorer l'efficacité dans le traitement des commandes et soigner l'interactivité avec les visiteurs du site.

Actuellement, les seuls points de contact entre la société et ses clients sont une série de boîtes aux lettres électroniques, chacune étant dédiée spécifiquement à un type particulier d'échange (commandes, avis personnels sur les produits, suggestions des clients). Le suivi des boîtes aux lettres et la prise en charge des commandes sont exclusivement réalisés par intervention humaine.

La taille relativement restreinte du catalogue des produits (une trentaine à ce jour) ne nécessite pas d'efforts démesurés de maintenance du site. Cet état de fait sera, bien entendu, remis en cause plus l'activité prendra de l'ampleur.

C'est dans cet esprit que les propriétaires ont décidé de migrer leur site vers une architecture dynamique qui leur permettra tant d'augmenter leur offre de produits sans rendre la maintenance du site ingérable, que d'automatiser le système de commandes et de faire

1. Le site peut être consulté depuis l'adresse suivante : http://www.info.fundp.ac.be/~jrm/LittleZon

évoluer la configuration au gré des tendances.

5.2 Travail préparatoire

Pour rappel, le travail préparatoire consiste, d'une part, à répertorier l'ensemble des pages du site, et, d'autre part, à transformer les sources *HTML* en documents *XML* bien formés.

5.2.1 Recensement des pages

Le recensement des pages consiste à répertorier dans un tableau l'adresse relative de chaque page par rapport à la racine du site. La figure 5.1 donne un aperçu de ce tableau pour notre cas.

URL-racine = http://www.info.fundp.ac.be/~jrm/LittleZon/
index.htm
CDs/C0001.htm
...(idem jusqu'à CDs/C0010.htm)
DVDs/D0001.htm
...(idem jusqu'à DVDs/D0014.htm)
Listes/ListCDs.htm
Listes/ListDVDs.htm
Listes/ListLivres.htm
Livres/L0001.htm
...(idem jusqu'à Livres/L0011.htm)
Nouveautes/NewCDs.htm
Nouveautes/NewDVDs.htm
Nouveautes/NewLivres.htm

Figure 5.1 - Tableau de recensement des pages du site.

5.2.2 Nettoyage du code *HTML*

Comme stipulé dans le chapitre relatif aux outils de support, cette étape est automatisée à l'aide de l'outil *TagSoup* qui se charge de transformer le code *HTML* source en un document *XML* bien formé.

Ainsi, les balises laissées ouvertes sont fermées, les guillemets manquants autour des valeurs d'attributs sont ajoutés et les balises inconnues du langage (ou non spécifiées par l'utilisateur) sont transformées en éléments vides.

5.3 Classification des pages

Le point d'entrée du site consiste en une page d'accueil (figure 5.2) qui introduit le visiteur à l'objet et au contenu du site et lui propose une colonne de navigation sur la gauche qui présente 7 hyperliens.

Figure 5.2 - La page d'accueil de *LittleZon* : «index.htm».

Du parcours en profondeur du site au départ de cette page, par activation des divers liens, nous élaborons une arborescence de navigation (figure 5.3).

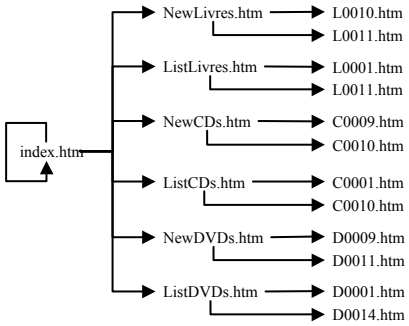


Figure 5.3 - Arborescence de navigation dans le site[2].

L'analyse de celle-ci, conjointement à l'étude des chemins d'accès aux différentes pages (cfr. le tableau de recensement des pages), va nous permettre d'élaborer une première classification des pages.

De ces deux sources d'information, nous pouvons déduire les éléments suivants :

- Les répertoires nommés *CDs*, *DVDs* et *Livres* contiennent les éléments terminaux de la figure 5.3. Il s'agit de l'ensemble des fiches produit. Les noms de ces éléments sont tous composés d'une lettre («C» pour un CD, «D» pour un DVD et «L» pour un livre) et

2. Notons que les 7 hyperliens présents sur la page d'accueil se retrouvent sur toutes les pages du site. Afin de préserver la lisibilité du schéma, nous n'avons noté leur présence que pour la page d'accueil.

d'une suite de quatre chiffres.

- Deux éléments sont dédiés à chaque type de produits dans la colonne de navigation. Il s'agit des liens *Nouveautés* et *Liste complète* qui renvoient à des pages contenues respectivement dans les répertoires *Nouveautes* et *Listes* au niveau de la structure des fichiers. Ces pages renvoient à leur tour vers des fiches produit des répertoires *DVDs*, *CDs* et *Livres*. Leur regroupement dans les dossiers *Nouveautes* et *Listes* porte à croire qu'il s'agit de pages de contenu comparable.
- La page d'accueil (*index.htm*) est unique en son genre. Nous la rangeons donc dans la catégorie des *pages orphelines*.

Une analyse visuelle plus détaillée des pages nous mène à confirmer les constatations faites ci-dessus. Nous notons toutefois une grande similitude entre les fiches produit des types *DVD* et *Livre*, ce qui nous permettrait d'en faire un type de pages unique.

Cependant, après réflexion, il nous semble plus raisonnable de les séparer, considérant la spécificité propre à chacun de ces produits et le souhait avoué d'évolution du site qui mènera peut-être à différencier le contenu des fiches dans l'avenir.

Nous obtenons ainsi cinq types de pages[3] et une page orpheline. Comme stipulé dans la description de la méthodologie, nous reprenons le tableau répertoriant les pages et y ajoutons une colonne qui spécifie le type de pages associé (figure 5.4).

URL-racine = http://www.info.fundp.ac.be/~jrm/LittleZon/	Type de pages
index.htm	orpheline
CDs/C0001.htm	CD
...(idem jusqu'à CDs/C0010.htm)	CD
DVDs/D0001.htm	DVD
...(idem jusqu'à DVDs/D0014.htm)	DVD
Listes/ListCDs.htm	ListeProduits
Listes/ListDVDs.htm	ListeProduits
Listes/ListLivres.htm	ListeProduits
Livres/L0001.htm	Livre
...(idem jusqu'à Livres/L0011.htm)	Livre
Nouveautes/NewCDs.htm	Nouveautes
Nouveautes/NewDVDs.htm	Nouveautes
Nouveautes/NewLivres.htm	Nouveautes

Figure 5.4 - Tableau associant chaque page à son type.

5.4 Enrichissement sémantique

Lors de cette étape, nous élaborons un fichier *XML* répondant au formalisme *Meta* (voir chapitre 3, section 3.5.1) pour chaque type de pages. Ce fichier décrit les concepts présents dans la page, leur structure et leur localisation dans l'arbre *HTML*.

Dans le cadre de ce document, nous allons détailler l'étude d'un seul type de pages : *Nouveautes*.

Nous commençons par effectuer l'analyse complète d'une page du type choisi au

3. Notons que les pages du type *ListeProduits* ne contient que des informations présentes dans d'autres pages. Elles ne seront donc pas traitées.

moyen de l'outil *Retrozilla*[4] (voir chapitre 4, section 4.2.1). De cette analyse, nous tirons un premier prototype du *Meta*.

Ensuite, nous confrontons ce dernier aux autres pages du type et y apportons les éventuelles modifications nécessaires. Nous obtenons alors un *Meta* qui valide toutes les pages.

5.4.1 Analyse d'une première page

- **Actions génériques**

Quel que soit le type de pages analysé, nous commençons toujours par effectuer quelques actions génériques.

Ouverture d'une page

Tout d'abord, nous ouvrons la page choisie dans l'éditeur *HTML* de *Mozilla* (figure 5.5). Le menu du plug-in *Retrozilla* apparaît à l'extrême droite de la barre des menus. La figure 5.6 présente les options de ce menu.

Figure 5.5 - La page «NewLivres.htm» vue dans l'éditeur *HTML* de *Mozilla*.

4. Le lecteur intéressé par les détails syntaxiques relatifs à l'élaboration d'un fichier *Meta* trouvera à l'Annexe B l'analyse du type de pages *DVD* entièrement réalisé à la main.

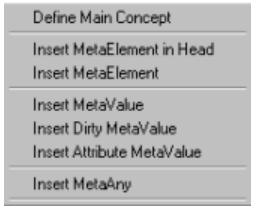


Figure 5.6 - Le menu de *Retrozilla*.

Déclaration du concept principal

La déclaration du concept principal se fait en choisissant l'option du menu *Define Main Concept*. Une boîte de dialogue (figure 5.7) s'ouvre. Nous introduisons le nom du concept principal dans la zone éditable *Name* et confirmons par le bouton *OK*.

Par convention, on donne au concept principal le nom du type de pages (ici *Nouveautes*).

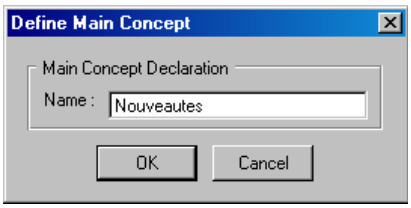


Figure 5.7 - Déclaration du concept principal *Nouveautes*.

Un type de pages n'ayant qu'un concept principal, cette action a pour effet de désactiver l'option *Define Main Concept* du menu.

Déclaration du titre de la page

Le titre d'une page *HTML* s'affiche dans la barre bleue au-dessus de la page. Souvent, il s'agit d'une donnée utile à extraire.

Pour cette raison, nous déclarons de manière générique un concept *TitrePage* qui contient cette donnée. Cela se fait en appelant l'option *Insert MetaElement in Head* dans le menu de *Retrozilla*.

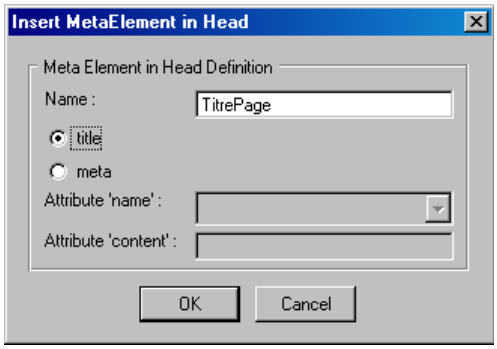


Figure 5.8 - Déclaration du concept *TitrePage* dans l'entête du code *HTML*.

Cette option permet de déclarer un concept dans l'entête du code *HTML* (partie non visible dans la fenêtre d'un navigateur). Elle appelle une boîte de dialogue (figure 5.8) qui réclame un nom pour le concept et le choix de l'élément à marquer dans l'entête (ici *title*). Cette action a pour effet de déclarer le concept et de marquer son contenu comme donnée à extraire.

Marquage des zones inintéressantes

La plupart des pages *HTML* contiennent des zones vides de contenu informationnel. Avant d'analyser le contenu proprement dit, nous marquons ces zones afin qu'elles soient ignorées dans la suite du processus.

Dans notre cas, nous notons quatre zones à ignorer :

- la colonne de navigation sur la gauche;
- le bandeau qui présente le nom du site («LittleZon»);
- la ligne «RetroWeb : Une étude de cas»;
- la fine ligne bleue en bas de la page.

Pour marquer une zone, nous y plaçons le curseur. La barre d'état affiche alors la hiérarchie des éléments *HTML* depuis la racine du contenu (<*body*>) jusqu'à la position du curseur. Nous sélectionnons dans la barre d'état la balise *HTML* dont le contenu doit être ignoré et appelons l'option du menu *Insert MetaAny*.

Un message nous confirme que le marquage a été réalisé et la zone est colorée en rouge. La figure 5.9 montre la fenêtre de l'éditeur après le marquage de la zone contenant le nom du site.

Figure 5.9 - Le contenu de l'élément *HTML <table>* sélectionné dans la barre d'état fait partie des zones à ignorer.

• Analyse du contenu

La déclaration des concepts

L'étude attentive de la page dans l'éditeur nous permet d'en découvrir la structure générale.

Le contenu proprement dit est composé d'un concept simple (le titre) et d'un tableau dont chaque cellule est une instance d'un concept composé (la description d'un produit).

La description d'un produit est à son tour composée d'une série de concepts simples :

- le nom du produit,
- une image cliquable,

- l'identité de l'auteur,
- une brève description,
- un paragraphe contenant un lien,
- un numéro de série facultatif.

La zone que recouvre un concept peut être sélectionnée de deux manières dans l'éditeur :

- Soit la zone est délimitée par un seul couple de balises. Alors la sélection s'opère de la même manière que pour les zones inintéressantes.
- Soit la zone est composée de plusieurs éléments *HTML* frères (ex: un titre et une série de paragraphes). Dans ce cas, on sélectionne la zone directement dans la fenêtre de l'éditeur au moyen de la souris.

Une fois la zone sélectionnée, on appelle l'option du menu *Insert MetaElement*. Cette action ouvre une boîte de dialogue qui permet de donner un nom au concept et de renseigner s'il est facultatif et/ou multivalué. Après confirmation, la zone marquée est colorée en jaune.

La figure 5.10 présente la boîte de dialogue lors de la déclaration du concept *Produit*. La figure 5.11 montre la fenêtre de l'éditeur après la déclaration du concept.

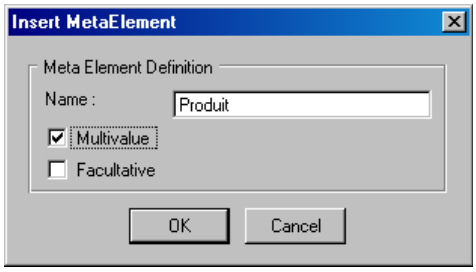


Figure 5.10 - Déclaration du concept *Produit*.

Figure 5.11 - La fenêtre de l'éditeur après la déclaration du concept *Produit*.

Nous déclarons de cette manière les concepts *TitreContenu*, *Produit*, *NomProduit*, *Image*, *Auteur*, *Description* et *NumeroSerie*. Notons deux cas particuliers :

1. Nous mentionnons le caractère multivalué du concept *Produit*. Cela nous évite de répéter les déclarations pour le second produit.
2. Nous mentionnons le caractère facultatif du concept *NumeroSerie*. En effet, ce concept est absent du second produit.

Ajoutons enfin que le lien présent dans le paragraphe de texte sous le titre «Pour en savoir plus...» renvoie à la même page que l'image cliquable. Il y a donc redondance. Le paragraphe ne contenant pas de données intéressantes, nous le marquons comme zone insignifiante (i.e. via l'option *Insert MetaAny*).

Le repérage des données

A chaque concept simple (i.e. ne contenant pas de sous-concepts) peut être associée une donnée à extraire. Par exemple la donnée du concept *Auteur* dans notre exemple est «Joanne-K. Rowling».

Pour rappel il existe trois types de données :

- un simple élément textuel (ex: le concept *NomProduit*)
- un texte contenant des balises *HTML* (ex: le concept *Description*)
- la valeur d'un attribut d'une balise *HTML* (ex: l'adresse source du concept *Image*)

La localisation d'une donnée simple se fait en sélectionnant une partie de l'élément textuel dans la fenêtre de l'éditeur. Ensuite on appelle l'option *Insert MetaValue*. Après confirmation, le texte est coloré en vert. Dans notre cas, nous marquons ainsi les données associées aux concepts *TitreContenu* et *NomProduit*.

Pour une donnée contenant des balises *HTML*, on utilise la même option du menu. La seule différence est que la zone sélectionnée doit couvrir l'ensemble des éléments qui font l'objet d'un formatage particulier. Nous marquons de la sorte la donnée associée au concept *Description*.

Afin de localiser une donnée qui est la valeur d'un attribut, on sélectionne la balise *HTML* qui contient cet attribut. Ensuite, on appelle l'option du menu *Insert Attribute MetaValue*. Cette action ouvre une boîte de dialogue qui nous permet de sélectionner dans une liste déroulante l'attribut qui nous intéresse. La figure 5.12 montre cette boîte de dialogue lors du marquage de la donnée associée au concept *Image* (valeur de l'attribut *src* de la balise *<img/>*). Dans notre cas, *Image* est le seul concept qui nécessite ce traitement.

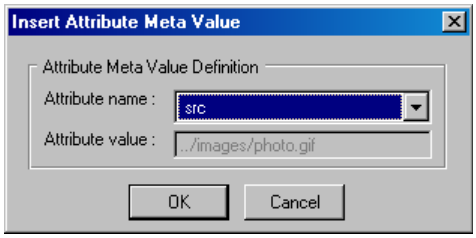


Figure 5.12 - La boîte de dialogue *Insert Attribute MetaValue*.

Notons encore que certaines données nécessiteront un post-traitement après leur extraction (voir chapitre 3, section 3.6.3). C'est le cas notamment de la donnée associée au concept *Auteur*. En effet, l'élément textuel contient à la fois le nom et le prénom de l'auteur. Nous donnons un marquage particulier à ce type de données en utilisant l'option du menu *Insert Dirty MetaValue* à la place de *Insert MetaValue*. Nous marquons de la sorte les concepts *Auteur* et *NumeroSerie*.

Ceci fait, nous avons terminé le traitement de la page *NewLivres.htm*. Nous enregistrons notre travail dans un fichier que nous appelons *Nouveautes.htm*[5].

Remarques complémentaires

La figure 5.13 montre la page *NewLivres.htm* une fois le traitement terminé.

Figure 5.13 - La page «NewLivres.htm» dans l'éditeur une fois le traitement terminé.

Nous remarquons que la seconde partie du contenu principal n'a fait l'objet d'aucun marquage. Cela est du au fait que le concept *Produit* est répétitif. Dans de tels cas, *Retrozilla* ne demande pas de répéter les manipulations, il suffit de marquer la première instance du concept et de préciser son caractère multivalué.

Notons également que certains éléments textuels n'ont pas été colorés en vert dans le contenu principal (les titres des paragraphes). Ils constituent les constantes de texte du type de pages.

5. Ce fichier peut être consulté à l'Annexe C, section C.1.

5.4.2 Génération de la description *Meta*

L'analyse de la page *NewLivres.htm* au moyen de *Retrozilla* nous a fourni un fichier *HTML* enrichi d'informations structurelles et sémantiques. Il nous faut maintenant transformer ce fichier en un document répondant au formalisme *Meta*.

Pour ce faire, nous utilisons l'outil *MetaGenerator* présenté au chapitre 4, section 4.2.2. Nous lui donnons en entrée le fichier *Nouveautes.htm* et il réalise la transformation automatiquement. Nous obtenons ainsi un nouveau fichier appelé *Nouveautes.xml*[6].

5.4.3 Validation des pages

L'étape précédente nous a fourni un fichier *Meta* qui décrit la page *NewLivres.htm*. Il faut maintenant voir si ce fichier décrit correctement les autres pages du type *Nouveautes*.

Pour ce faire, nous utilisons l'outil *HTMLValidator* présenté au chapitre 4, section 4.2.3.

Nous lui donnons en entrée le fichier *Meta* et l'ensemble des pages du type *Nouveautes*.

Validation de la page *NewDVDs.htm*

Figure 5.14 - Un des produits de la page «NewDVDs.htm» vue dans l'éditeur de *Mozilla*.

Lors de l'essai de validation du fichier *NewDVDs.htm*, l'outil délivre le message suivant :

```
-----------------------------------------------------
HTML Father : h3
HTML Position : 1
META : #text
Auteur
Réalisateur
fin Auteur
nbre element trouvé : 0 --> incorrect
fin Produit
nbre element trouvé : 0 --> incorrect
NewDVDs.htm n'est pas conforme au META
-----------------------------------------------------
```

Ce message nous indique que dans le concept *Auteur*, le fichier *Meta* attendait la constante de texte «Auteur» et a rencontré à la place «Realisateur» (voir figure 5.14).

Nous devons modifier le fichier *Meta* pour tenir compte de cette différence.

Pour ce faire, nous définissons deux nouveaux concepts : *Personnalite* et *Realisateur*.

Realisateur a la même structure que *Auteur* (excepté la constante textuelle) et *Personnalite* contient une structure de choix : une *Personnalite* est soit un *Auteur*, soit un *Realisateur*. La figure 5.15 illustre graphiquement la structure créée pour le concept *Personnalite*.

6. Ce fichier peut être consulté à l'Annexe C, section C.2.

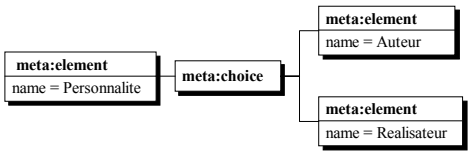


Figure 5.15 - Structure de choix pour le concept *Personnalite*.

Une fois cette modification faite, la validation réussit pour la page *NewDVDs.htm*.

Validation de la page *NewCDs.htm*

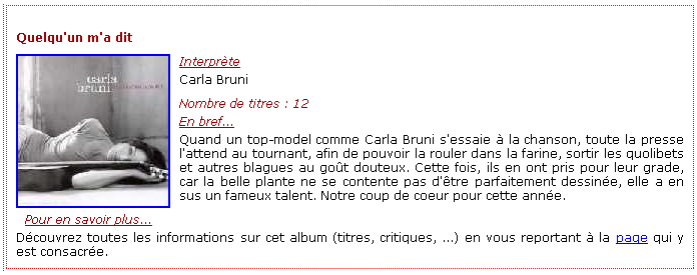


Figure 5.16 - Un des produits de la page «NewCDs.htm» vue dans l'éditeur de *Mozilla*.

Lors de l'essai de validation du fichier *NewCDs.htm*, le même problème survient. Cette fois la constante de texte est «Interprète» (voir figure 5.16).

Pour résoudre ce problème, nous ajoutons simplement une nouvelle branche à la structure de choix dans le concept *Personnalite* et déclarons un nouveau concept *Interprete*.

La validation suivante échoue à nouveau. Cette fois nous recevons le message suivant :

```
--------------------------------
HTML Father : h3
HTML Position : 1
META : #text
En bref...
Nombre de titres : 12
fin Description
nbre element trouvé : 0 --> incorrect
fin Produit
nbre element trouvé : 0 --> incorrect
NewCDs.htm n'est pas conforme au META
--------------------------------
```

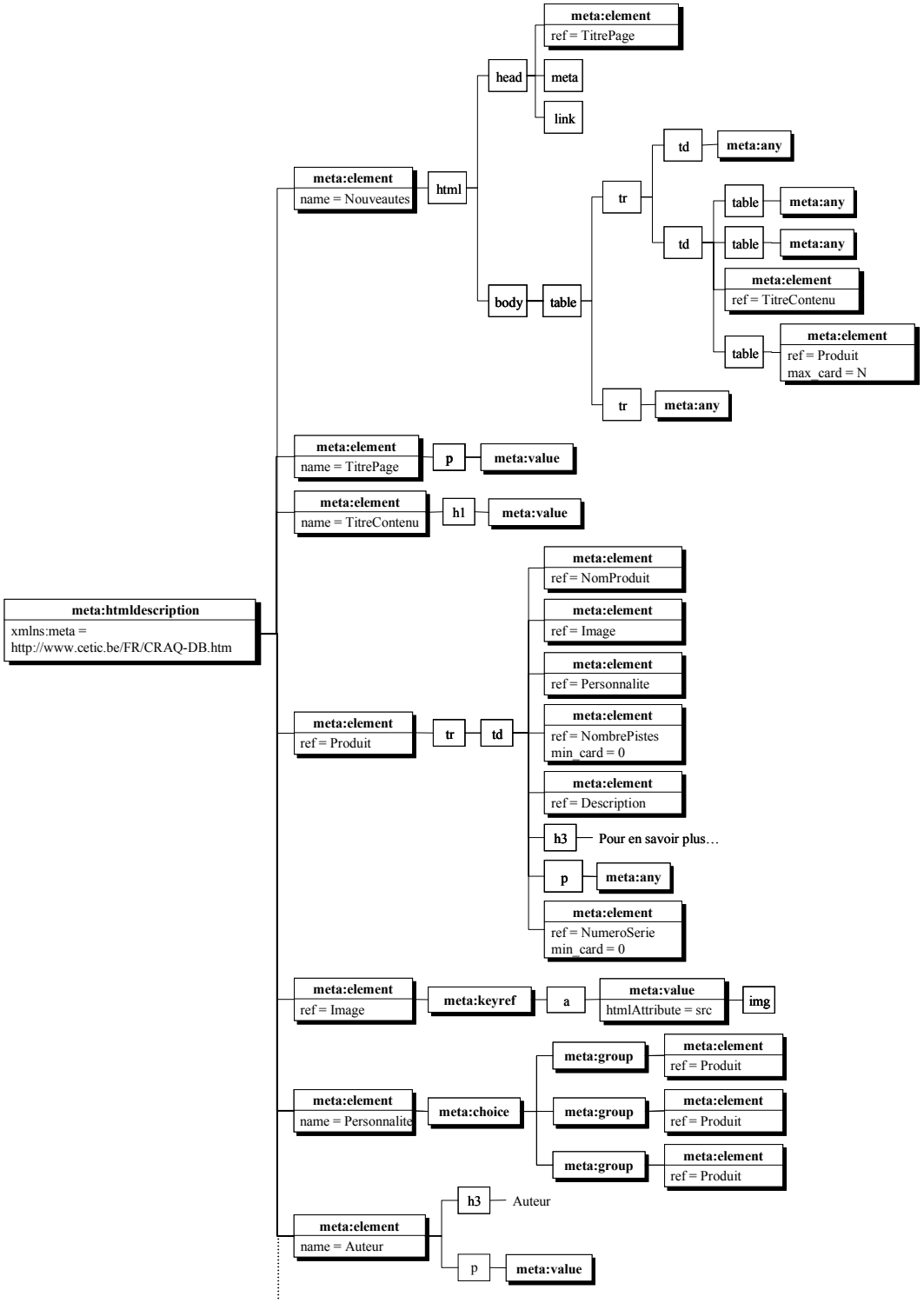
Ce message nous indique que là où le *Meta* attendait la constante de texte «En bref...», il a rencontré la chaîne de caractères «Nombre de titres : 12».

Nous consultons la page *NewCDs* dans l'éditeur de *Mozilla* et nous remarquons qu'une nouvelle information est insérée entre les concepts *Personnalite* et *Description*.

Pour régler ce problème, nous déclarons un nouveau concept *NombrePistes* et le référençons en tant qu'élément facultatif dans la description du concept *Produit*, entre *Personnalite* et *Description*. Seule la donnée chiffrée mérite d'être extraite; aussi nous marquons la donnée associée au nouveau concept de l'attribut *dirty*.

Ceci fait, la validation réussit. Nous avons maintenant un fichier *Meta* exhaustif pour toutes les pages du type *Nouveautes*[7]. La figure 5.17 illustre graphiquement le *Meta* complet.

7. Ce fichier peut être consulté à l'Annexe C, section C.3.


meta:htmldescription
xmlns:meta = http://www.cetic.be/FR/CRAQ-DB.htm
meta:element
name = Nouveautes
html
head
meta:element
ref = TitrePage
meta
link
body
table
tr
td
meta:any
td
table
meta:any
table
meta:any
meta:element
ref = TitreContenu
table
meta:element
ref = Produit
max_card = N
tr
meta:any
meta:element
name = TitrePage
p
meta:value
meta:element
name = TitreContenu
h1
meta:value
meta:element
ref = Produit
tr
td
meta:element
ref = NomProduit
meta:element
ref = Image
meta:element
ref = Personnalite
meta:element
ref = NombrePistes
min_card = 0
meta:element
ref = Description
h3
Pour en savoir plus…
p
meta:any
meta:element
ref = NumeroSerie
min_card = 0
meta:element
ref = Image
meta:keyref
a
meta:value
htmlAttribute = src
img
meta:element
name = Personnalite
meta:choice
meta:group
meta:element
ref = Produit
meta:group
meta:element
ref = Produit
meta:group
meta:element
ref = Produit
meta:element
name = Auteur
h3
Auteur
p
meta:value

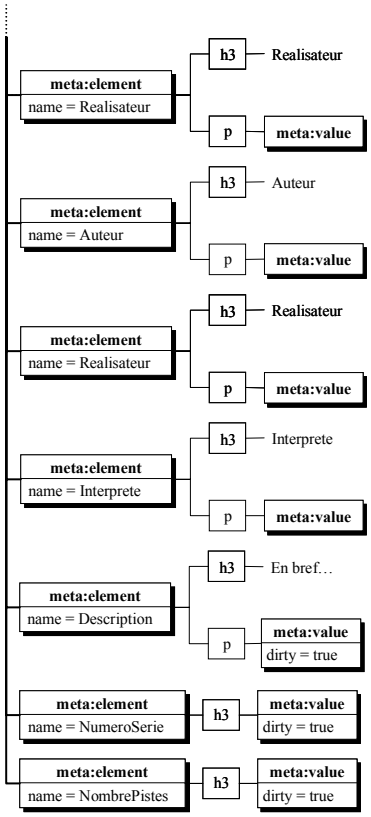


Figure 5.17 - Représentation graphique du *Meta* complet pour le type de pages *Nouveautes*.

5.5 Extraction

5.5.1 Le schéma physique

L'objectif de cette étape est d'obtenir un schéma physique de la structure de données du type de pages.

Ce processus est automatisé grâce à l'outil *SchemaExtractor* présenté au chapitre 4, section 4.3.1.

Nous lui fournissons en entrée le fichier *Meta* et il génère le *XML Schema* correspondant, conformément aux règles de traduction définies au chapitre 3, section 3.6.1.

La figure 5.18 propose une représentation graphique de la structure de données du type de pages *Nouveautes*[8] produite par l'outil *SchemaExtractor*.

8. Le code *XML Schema* correspondant peut être consulté à l'Annexe D, section D.1.

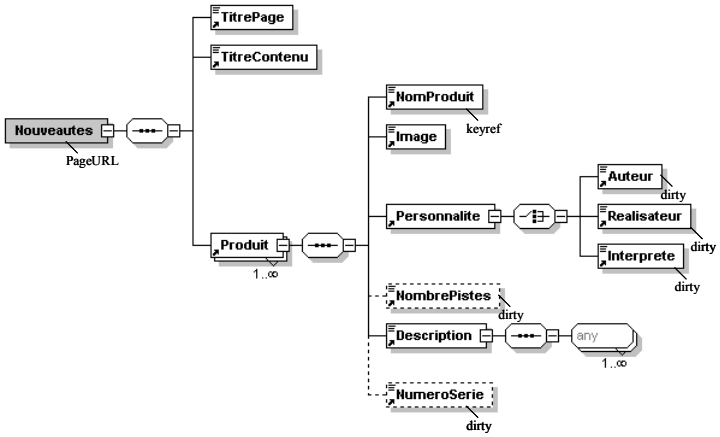


Figure 5.18 - Structure de données du type de pages *Nouveautes*.

5.5.2 Les données

Lors de cette étape, nous extrayons les données utiles des pages *HTML* classées dans la catégorie analysée. Ces données sont produites dans un fichier *XML* conforme au *XML Schema* obtenu précédemment.

Ce travail est automatisé par l'outil *DataExtractor* décrit au chapitre 3, section 4.3.2.

Nous proposons ci-dessous un extrait du fichier *XML*[9] contenant les données de la page *NewLivres.htm*. Il s'agit des données relatives au premier produit présenté sur la page.

```
<Produit>
    <NomProduit>Harry Potter et l'Ordre du Phoenix</NomProduit>
    <Image keyref="../Livres/L0010.htm">../images/smallimgL0010.jpg</Image>
    <Personnalite>
        <Auteur dirty="true">Joanne-K. Rowling</Auteur>
    </Personnalite>
    <Description>
        Probablement une des sorties les plus attendues de l'histoire de la littérature...
        voilà enfin en version française le 5ème volume des aventures du plus célèbre
        des apprentis sorciers. Plus de 900 pages de plaisir et de suspense qui vous feront
        découvrir les premiers pas de <i>Harry</i> dans l'adolescence. Ca valait la peine
        d'attendre.
    </Description>
    <NumeroSerie dirty="true">Numéro dans la série : 5</NumeroSerie>
</Produit>
```

5.5.3 Nettoyage des données et adaptation du schéma physique

Lors de cette étape, nous traitons les éléments marqués de l'attribut *dirty*. Cela se fait au moyen des outils *DataCleaner* et *SchemaAdaptor* présentés au chapitre 4, section 4.3.3.

Dans le type de pages *Nouveautes*, nous en recensons cinq : *NombrePistes*, *NumeroSerie, Auteur*, *Realisateur* et *Interprete*.

9. Le fichier de données complet pour le type de pages *Nouveautes* peut être consulté à l'Annexe D, section D.2.

NombrePistes et *NumeroSerie*

Les éléments *NombrePistes* et *NumeroSerie* ont été marqués pour la même raison : la donnée utile n'est qu'une partie de la donnée extraite.

Dans les deux cas, la sous-chaîne de caractères à éliminer est une constante qui débute la donnée textuelle. Nous pouvons donc spécifier à l'outil *DataCleaner* l'endroit de la chaîne à partir duquel il faut conserver les informations.

A titre d'exemple, nous proposons ci-dessous un élément *XML* avant et après le traitement.

```
Avant le traitement : <NumeroSerie dirty="true">Numéro dans la série : 5</NumeroSerie>
Après le traitement :  <NumeroSerie>5</NumeroSerie>
```

Une fois ces modifications réalisées, l'outil *DataCleaner* passe la main au *SchemaAdaptor* qui se charge de supprimer l'attribut *dirty* dans les concepts traités.

Auteur, *Realisateur* et *Interprete*

Les données extraites pour les concepts *Auteur*, *Realisateur* et *Interprete* présentent la même anomalie. Ces données contiennent les valeurs respectives de deux sous-concepts (*Prenom* et *Nom*) séparés par un espace.

Nous spécifions à l'outil *DataCleaner* qu'il s'agit d'une donnée composée et lui précisons que les sous-données sont séparées par espace. Il crée deux sous-éléments au concept composé, chacun contenant une partie de la donnée de départ.

A titre d'exemple, nous proposons ci-dessous un élément *Auteur* avant et après le traitement.

```
Avant le traitement : <Auteur dirty="true">Joanne-K. Rowling</Auteur>
Après le traitement : <Auteur>
                              <AuteurChild_1>Joanne-K.</AuteurChild_1>
                              <AuteurChild_2>Rowling</AuteurChild_2>
                      </Auteur>
```

Ceci fait, le *DataCleaner* passe le relais au *SchemaAdaptor* qui apporte les modifications nécessaires au *XML Schema* de départ.

La figure 5.19 présente graphiquement la structure *XML Schema* après les modifications[10].

10. Le fichier *XML schema* et le fichier de données définitifs peuvent être consultés à l'Annexe D, sections D.3 et D.4.

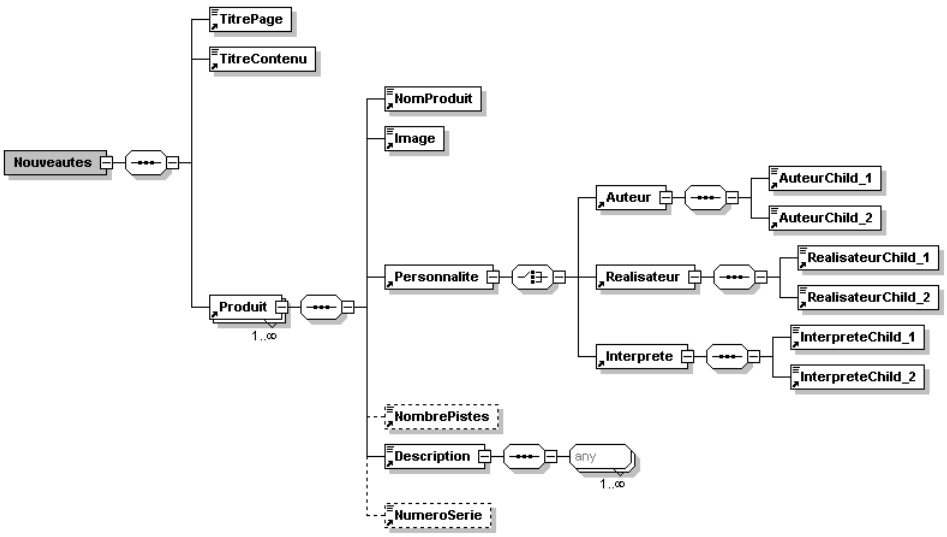


Figure 5.19 - Représentation graphique du *XML Schema* définitif.

5.6 Intégration et conceptualisation

Lors de cette étape, nous allons tout d'abord importer les structures *XML Schema* dans *DB-Main*. Ensuite, nous supprimons les constructions propres à la norme *XML Schema* et obtenons un type d'entités par type de pages. Nous opérons des transformations sur ces types d'entités pour créer un schéma conceptuel par type de pages. Ceci fait, nous intégrons les différentes structures pour obtenir un schéma conceptuel de l'ensemble du domaine d'application.

5.6.1 Importation des structures *XML Schema* dans *DB-Main*

L'outil *CASE DB-Main* (voir chapitre 4, section 4.4) dispose d'un module permettant de transformer un modèle de document *XML* en un schéma *GER*. La figure 5.20 montre la structure de données du type de pages *Nouveautes* après l'importation.

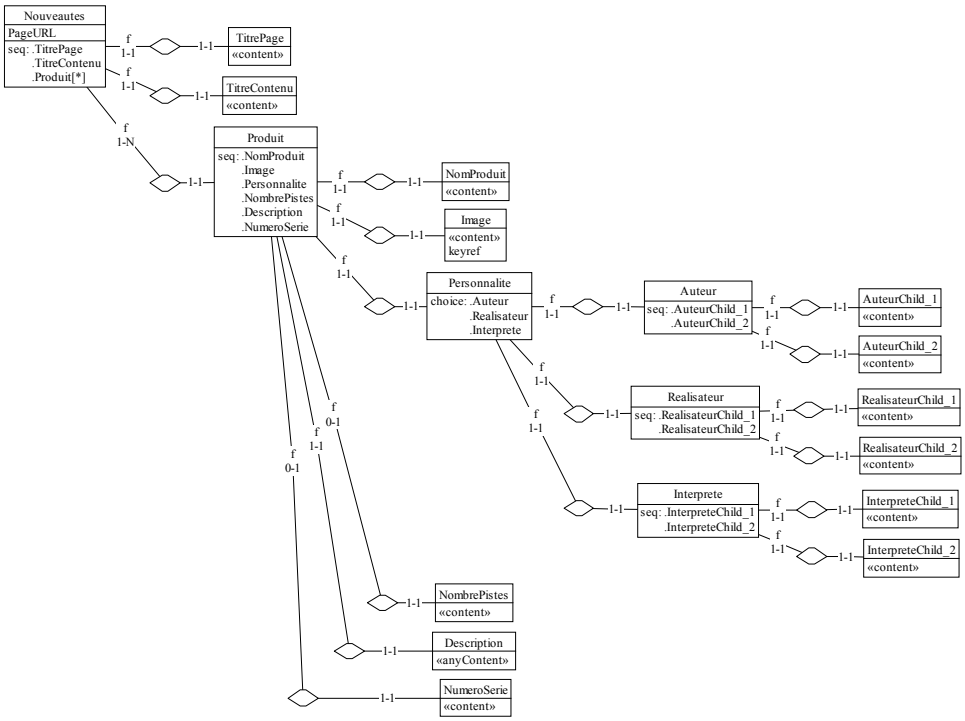


Figure 5.20 - Le type de pages *Nouveautes* dans un schéma *GER* conforme au modèle *XML Schema*.

5.6.2 Suppression des constructions propres aux *XML Schemas*

Pour rappel, cette étape consiste à appliquer les manipulations décrites au chapitre 3, pages 58 à 61, à savoir :

- supprimer la structure hiérarchique;
- supprimer les contraintes *seq*;
- transformer les contraintes *choice* en contraintes *exact-1* ou *excl.*

En plus de cela, nous donnons des noms représentatifs aux concepts issus du processus de nettoyage des données (section 5.5.3). En effet, les noms générés automatiquement sont peu intuitifs et méritent donc d'être clarifiés.

Ces manipulations sont appliquées sur les quatre types de pages. Etant donné que cette étape consiste à appliquer de manière répétitive des transformations décrites précédemment, nous nous contentons de fournir le schéma représentant les quatre types d'entités après ce processus (figure 5.21).

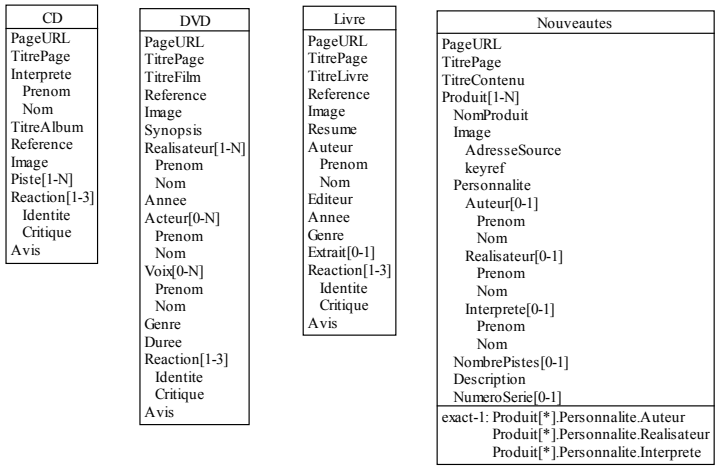


Figure 5.21 - Les quatre types de pages après suppression de la hiérarchie et des contraintes propres au modèle *XML Schema*

5.6.3 Conceptualisation des différents types d'entités

A des fins de lisibilité et pour faciliter l'intégration entre les différents types d'entités, nous opérons certaines transformations sur le schéma obtenu à l'étape précédente. Ainsi, on va notamment transformer les attributs multivalués en types d'entités, désagréger certains attributs composés, etc.

Nous n'allons pas rentrer dans les détails de toutes les manipulations effectuées avant l'intégration. Nous nous contentons de donner quelques transformations appliquées au type d'entités *Nouveautes*.

- L'attribut *Produit* est composé de sous-éléments qui sont parfois eux-mêmes composés. Un *Produit* existe indépendamment du fait qu'il soit une nouveauté ou pas. Nous transformons dont l'attribut *Produit* en un type d'entités.
- L'attribut *Personnalite* du type d'entités *Produit* a également une structure complexe. Pour les mêmes raisons que dans le cas précédent, nous le transformons également en un type d'entités à part entière.
- Le nouveau type d'entités *Personnalite* contient trois attributs qui ont la même structure et sont mutuellement exclusifs : *Auteur*, *Realisateur* et *Interprete*. Ceci cache une relation de généralisation/spécialisation (ou relation *is-a*) entre le type d'entités et ces trois attributs. Cette relation signifie qu'une personnalité est soit un auteur, soit un réalisateur, soit un interprète. On transforme donc ces trois attributs en type d'entités et celles-ci deviennent des sous-types du type d'entités *Personnalite*.
- L'attribut *Image* du type d'entités *Produit* est lui-même composé de deux attributs : *AdresseSource* et *keyref*. En réfléchissant bien, on peut considérer que l'attribut *keyref* est plutôt caractéristique du produit que de l'image. En effet, cet attribut matérialise un lien qui, dans la page, renvoie vers la fiche complète du produit présenté. Nous choisissons donc de désagréger l'attribut *Image* en deux attributs atomiques.

Nous effectuons une analyse comparable des autres types d'entités et obtenons le schéma de la figure 5.22

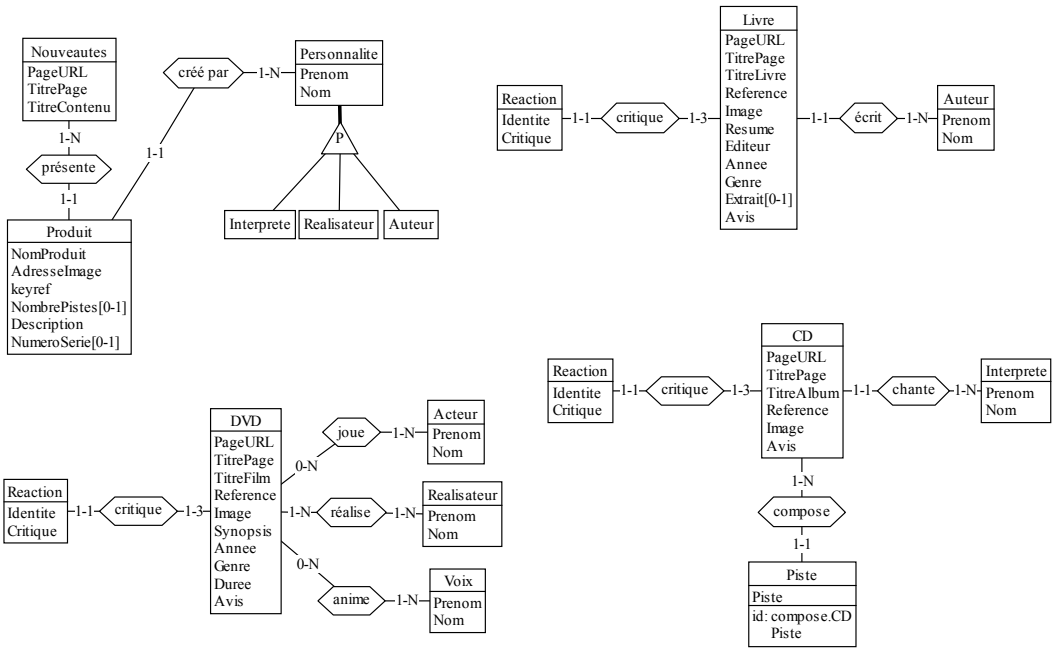


Figure 5.22 - Les quatre types de pages après un premier traitement.

5.6.4 Intégration des structures

Pour rappel, l'intégration des structures consiste en trois points (voir chapitre 3, pages 61 à 62) :

- identifier et intégrer les types d'informations comparables;
- expliciter les liens entre les différents types;
- éliminer les redondances.

Pour ce faire, on analyse les schémas conceptuels et on vérifie nos constatations en consultant des pages du site.

• Les types d'informations comparables

Constatations

1.Trois types d'entités (*CD*, *DVD* et *Livre*) ont un certain nombre d'attributs communs :

- l'adresse de la page (attribut *PageURL*);
- le titre de la page (attribut *TitrePage*);
- un nom de produit (attributs *TitreCD*, *TitreFilm* et *TitreLivre*);
- une référence (attribut *Reference*);

- une image (attribut *Image*);
- un avis (attribut *Avis*);
- les attributs *Genre* et *Annee* sont communs aux types d'entités *DVD* et *Livre*.

2.Chacun de ces trois types d'entités est relié par un type d'associations à un type d'entités appelé *Reaction*.

3. Ces mêmes types d'entités sont reliés par des types d'associations à d'autres types d'entités de noms différents mais de même structure. En effet, les types d'entités *Auteur*, *Interprete*, *Realisateur*, *Acteur* et *Voix* sont tous composés d'un prénom et d'un nom.

Consultation des pages

La consultation de quelques pages confirme nos impressions.

1.Les caractéristiques communes ont la même sémantique si on considère que les trois types d'entités sont des spécialisations d'un type plus général.

2.Les trois types d'entités appelés *Reaction* représentent le même type d'informations.

3.Chacun des types d'entités composés d'un nom et d'un prénom représentent une personnalité qui joue un rôle vis-à-vis du type de produit auquel il est relié.

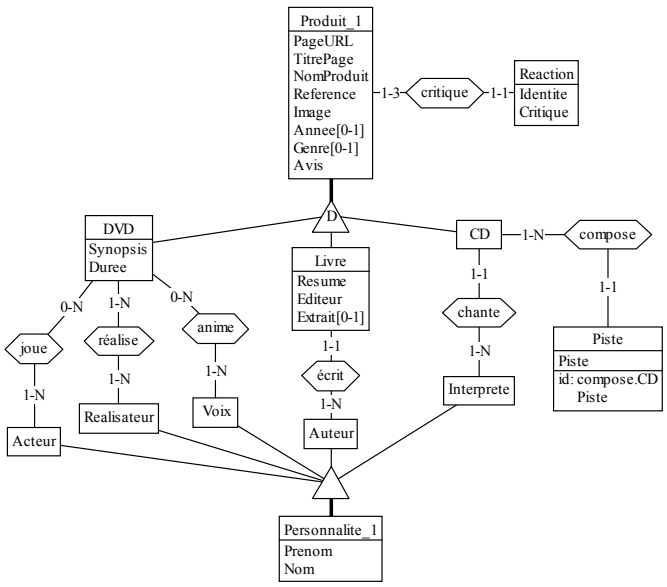


Figure 5.23 - Schéma conceptuel intégré des trois types de produits.

Actions

1.Nous créons un nouveau type d'entités (*Produit_1*[11]) qui devient le sur-type des trois types d'entités *CD*, *DVD* et *Livre*. Le lien entre le sur-type et les sous-types est soumis à une contrainte de disjonction. Celle-ci signifie qu'un *Produit_1* ne peut être à la fois plusieurs de ses sous-types.

11. On ajoute un indice au type d'entités car un autre porte déjà ce nom.

Nous transférons toutes les caractéristiques communes dans le nouveau type d'entités.

Les deux attributs (*Genre* et *Annee*) qui n'apparaissent que dans deux des trois types sont également transférés, mais ils deviennent facultatifs. Ceci permet d'envisager à l'avenir l'ajout de ces attributs pour les produits de type *CD*.

2.Les trois types d'entités appelés *Reaction* sont fusionnés.

3.Nous créons un sur-type commun aux types d'entités *Auteur*, *Interprete*, *Groupe*, *Acteur* et *Voix*. Nous y transférons les deux attributs communs.

La figure 5.23 montre la partie de schéma influencée par ces transformations.

• Les liens

Constatations

On recense un seul lien dans le schéma. Il est matérialisé par l'attribut *keyref* du type d'entités *Produit*.

Consultation des pages

Selon les pages le lien référence l'adresse d'une page de type *CD*, *DVD* ou *Livre*.

Par conséquent, le domaine de valeurs de l'attribut *keyref* est inclus dans le domaine de valeurs de l'attribut *PageURL* du type d'entités *Produit_1*.

Actions

Par construction, l'adresse d'une page est unique. Nous faisons donc de l'attribut *PageURL* un identifiant du type d'entités *Produit_1*.

Ceci nous permet de déclarer une contrainte référentielle entre l'attribut *keyref* du type d'entités *Nouveaute* et l'attribut *PageURL* du type d'entités *Produit_1*.

• Les informations redondantes

Constatations

Suite aux transformations précédentes, des types d'entités ont le même nom (ou presque) :

1.*Produit* et *Produit_1*,

2.*Personnalite* et *Personnalite_1*.

Consultation des pages

1.Les instances du type d'entités *Produit* sont un sous-ensemble des instances du type d'entités *Produit_1*. En effet, le premier représente les instances du second qui font partie des nouveaux produits présentés sur le site.

Deux attributs du type d'entités *Produit* sont caractéristiques d'un seul sous-type du type d'entités *Produit_1* :

- *NumeroSerie* ne s'adresse qu'aux produits de type *Livre*;
- *NombrePistes* ne s'adresse qu'aux produits de type CD.

Deux autres attributs (*NomProduit* et *AdresseImage*) sont redondants par rapport aux attributs *NomProduit* et *Image* du type d'entités *Produit_1*.

2.Toutes les informations contenues dans les instances du type d'entités *Personnalite* se retrouvent dans des instances du type d'entités *Personnalite_1*. Il s'agit des noms et prénoms des auteurs, réalisateurs et interprètes qui sont reliés à un nouveau produit.

Actions

1. L'attribut *NumeroSerie* du type d'entités *Produit* est transféré dans le type d'entités *Livre*.

L'attribut *NombrePistes* du type d'entités *Produit* est supprimé. En effet, cette information peut-être retrouvée grâce à la relation entre *CD* et *Pistes*.

Les attributs *NomProduit* et *AdresseImage* du type d'entités *Produit* sont supprimés. En effet, étant redondants par rapport aux attributs *NomProduit* et *Image* du type d'entités *Produit_1*, ils peuvent être retrouvés via la contrainte référentielle entre les deux types d'entités.

2.Le type d'entités *Personnalite* et ses sous-types sont supprimés du schéma. Les informations qu'ils représentent peuvent également être retrouvées via la contrainte référentielle.

• Manipulations complémentaires suite à la consultation des pages

La consultation attentive des pages nous a fourni d'autres informations :

- Les attributs *TitrePage* et *NomProduit* dans les pages de type *Produit_1* ont systématiquement la même valeur. Il en va de même pour les attributs *TitrePage* et *TitreContenu* dans les pages de type *Nouveaute*.
  Action : nous supprimons les attributs *TitrePage*.
- L'attribut *Reference* du type d'entités *Produit_1* correspond au nom de fichier (moins l'extension) de l'attribut *PageURL* dans le même type d'entités. Il est composé d'une lettre et de quatre chiffres. La lettre correspond à la première lettre du type des produits présentés : «C» pour un produit *CD*, «D» pour un produit *DVD*, «L» pour un produit *Livre*.
  Action : l'attribut *Reference* devient un identifiant du type d'entités *Produit*.
- Le type d'entités *Produit_1* a maintenant deux identifiants. Etant donné que la valeur d'instance de l'identifiant *Reference* correspond à une partie de la valeur d'instance de l'identifiant *PageURL*, ce dernier n'est pas un identifiant minimal. Il semble donc inutile de conserver *PageURL*. Cependant, ce dernier est la cible d'une contrainte référentielle. Si on veut le supprimer, il faut donc procéder à un traitement des fichiers de données pour que les valeurs d'instance de l'attribut *keyref* soient réduites au nom de fichier moins l'extension.
  Action : on supprime l'attribut *PageURL* du type d'entités *Produit_1*, on transfère la cible de la contrainte référentielle vers l'identifiant *Reference* et on traite les fichiers de données.

Après ces manipulations, il ne reste plus dans le type d'entités *Nouveaute*s que les attributs *PageURL*, et *TitreContenu*.

Toutes les informations nécessaires à construire une page de type *Nouveautes* sont disponibles dans les autres types d'entités. Aussi décidons-nous de supprimer le type d'entités *Nouveautes* du schéma.

5.6.5 Normalisation conceptuelle

Les manipulations précédentes sont indispensables mais ne suffisent pas à rendre le schéma normalisé.

Pour rappel la normalisation consiste à donner une apparence définitive au schéma conceptuel. Il reste à appliquer certaines transformations pour améliorer notamment la lisibilité et l'expressivité (remplacement de certains noms, transformation des contraintes référentielles en types d'associations, adaptation de certaines cardinalités, etc).

Dans notre cas nous réalisons les manipulations suivantes :

- remplacement de noms : *Produit* devient *NouveauProduit*, *Produit_1* devient *Produit* et *Personnalite_1* devient *Personnalite.*
- extension de la cardinalité du rôle *critique.Produit* de [1-3] à [0-N];
- transformation de la contrainte référentielle en un type d'associations;
- ajout d'un identifiant aux types d'entités *Reaction*, *Personnalite* et *Piste;*
- changement de la cardinalité des rôles joués par les types d'entités *Auteur*, *Realisateur*, *Interprete*, *Acteur* et *Voix* de [1-N] à [0-N].

La figure 5.24 présente le schéma conceptuel complet et définitif.

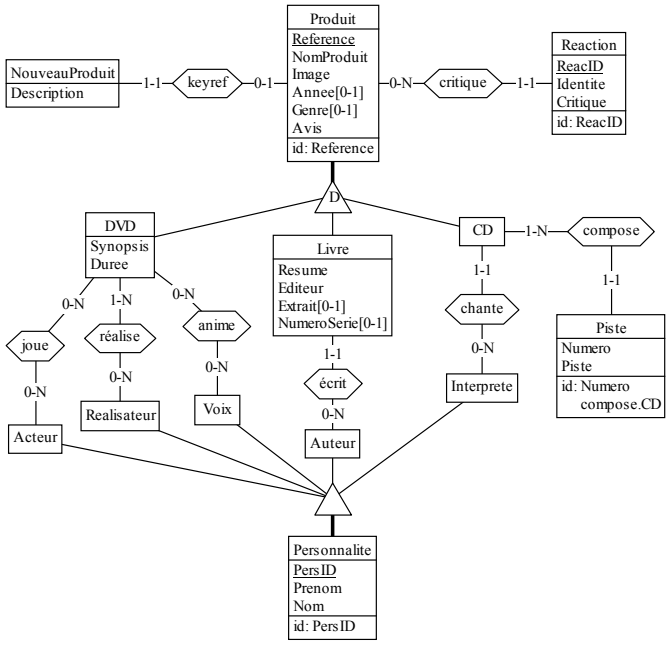


Figure 5.24 - Schéma conceptuel complet du domaine d'application

5.7 Conception de la base de données

Dans cette étape, nous générons au départ du schéma conceptuel les structures physiques nécessaires pour accueillir les données extraites des pages *HTML*.

C'est ici qu'il faut faire le choix de la technologie d'implémentation. Pour notre cas, nous allons créer une base de données sur le modèle relationnel.

Ce processus est entièrement automatisé par l'outil CASE *DB-Main*. Nous lui demandons d'abord de transformer le schéma conceptuel en structures relationnelles[12] (figure 5.25) et ensuite de générer le code *DDL* pour la création de la base de données[13].

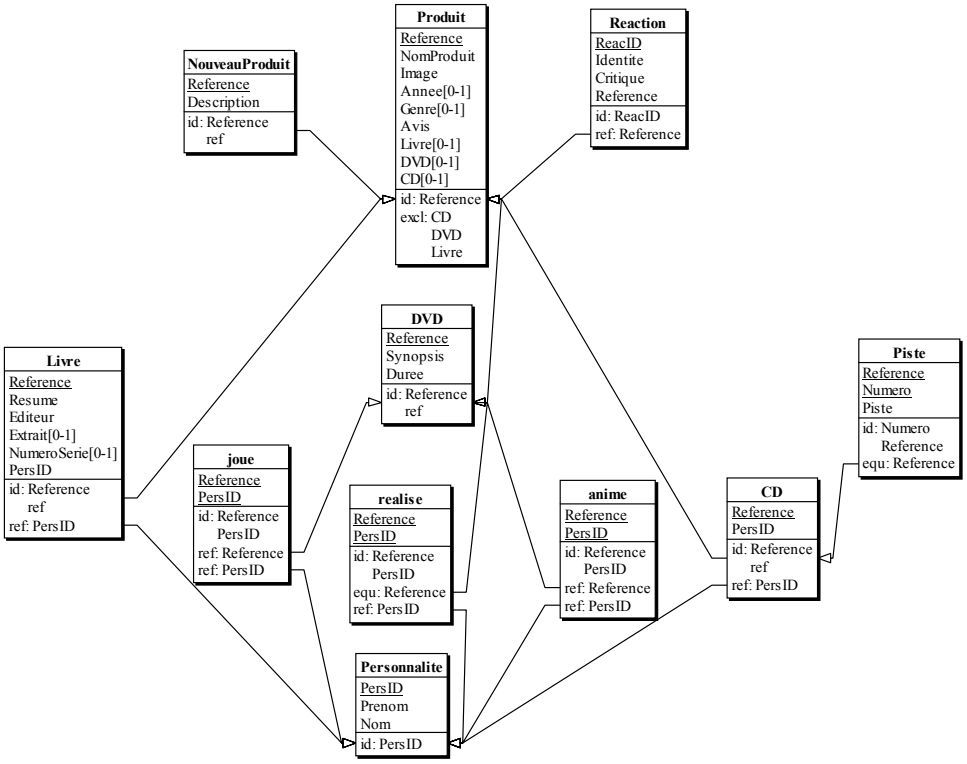


Figure 5.25 - Schéma relationnel généré par *DB-Main*

5.8 Migration des données

DB-Main permet d'associer un identifiant d'objet aux types d'entités, types d'associations et attributs. Cet identifiant reste associé à l'objet tout au long des transformations.

DB-Main permet également d'enregistrer dans un journal toutes les transformations réa-

12. Pour simplifier l'illustration, nous avons fusionné les sous-types d'entités de *Personnalité* dans leur sur-type.
13. Ce code peut être consulté à l'Annexe E.

lisées sur un schéma, de la conceptualisation à la conception.

Sur la base de ces deux sources d'informations, *DB-Main* génère un migrateur, i.e. un programme qui va appliquer des règles de traduction aux fichiers *XML* de données pour les faire correspondre à la structure de la base de données et les y enregistrer automatiquement.

5.9 Exemple d'amélioration du site

Suite à la migration des informations dans une base de données, les pages du site peuvent être reconstruites dynamiquement.

La reconstruction du site peut simplement consister à reproduire à la demande les pages telles qu'elles étaient auparavant, sans ajouter de fonctionnalités supplémentaires.

Souvent, cependant, les propriétaires du site en profitent pour augmenter les possibilités. Dans notre cas, on peut imaginer qu'ils décident d'ajouter à leur configuration un module de recherche qui permet au visiteur de sélectionner les informations sur la base de certains critères.

On voit ainsi apparaître (figure 5.26) une zone de type formulaire dans une colonne à droite de la page. Celle-ci permet de rechercher des produits sur la base d'un titre, d'un nom de personne, d'un genre ou d'une année. La page donnée en exemple montre les résultats d'une recherche qui demande tous les produits dont le titre contient «Harry Potter».

Figure 5.26 - Une page du site construite sur la base d'un critère de recherche.

Chapitre 6

Conclusion

Nous avons présenté dans ce document une méthodologie permettant d'extraire de pages *HTML* les données et leur structure sémantique.

Appliquée sur une étude de cas, la méthodologie a montré son efficacité en fournissant des résultats satisfaisants. Cependant, certains points méritent d'être améliorés!

Classification des pages

La classification des pages basée sur l'arborescence de navigation et l'étude des chemins d'accès aux fichiers est efficace en termes de résultats, mais coûteuse en temps. En effet, si pour un nombre de pages relativement réduit, cette approche est réaliste, elle devient laborieuse pour des sites de grande taille.

L'automatisation de cette tâche peut être réalisée en appliquant des techniques de *clustering*. Cette approche consiste à comparer automatiquement les pages source sous l'angle de leur syntaxe et à les regrouper en classes en fonction de leur distance syntaxique.

L'utilisation de telles techniques permettrait un gain de temps considérable pour l'analyse de sites de grande taille. En effet, le *clustering* nous fournirait rapidement une première classification que nous pourrions ensuite affiner par une analyse visuelle des pages.

Enrichissement sémantique

L'outil *Retrozilla* utilisé pour l'enrichissement sémantique des pages *HTML* fait preuve d'un grand confort visuel et donne des résultats satisfaisants.

Cependant, comme nous l'avons signalé au chapitre 4, section 4.2.1, cet outil ne permet pas de modifier la description *Meta* lors de la découverte de différences entre plusieurs pages d'un même type.

Pour remédier à ce problème, on pourrait imaginer améliorer l'outil de validation des pages (i.e. *HTMLValidator*) pour qu'il propose au fur et à mesure des échecs de validation les solutions possibles (ajout d'un concept, création d'une structure de choix, ajout du caractère optionnel ou multivalué à un concept) pour prendre en compte les différences constatées et les intégrer à la description *Meta*. Cette solution est actuellement à l'étude au *CETIC*.

Pour améliorer la facilité d'analyse d'une page, on pourrait également intégrer à *Retrozilla* un

module qui proposerait des interprétations possibles des structures *HTML* présentes dans la page. Ces propositions se baseraient sur les éléments mis en évidence au chapitre 2, section 2.1.2.

Les outils Java

Les outils *Java* utilisés pour générer le *Meta*, valider les pages, extraire les données et les *XML schemas* et nettoyer les données sont actuellement exécutés en ligne de commande.

Afin d'en faciliter l'utilisation, il conviendrait de développer une interface graphique qui permettrait d'appeler chacun de ces outils au gré des besoins.

***La traduction des* XML Schemas *en schémas* GER**

Comme nous l'avons signalé au chapitre 4, section 4.4, *DB-Main* dispose d'un importateur de *DTDs*, mais pas de l'équivalent pour les *XML Schemas*.

La traduction d'un *XML Schema* en schéma *GER* doit donc actuellement se faire à la main ou via une transformation transitoire en *DTD*, ce qui est source d'erreurs. En effet, l'impossibilité de représenter certaines structures propres au *XML Schemas* dans le modèle *DTD* nécessite des adaptations manuelles.

Par conséquent, il nous semble important d'envisager le développement d'un importateur spécifiquement adapté à la norme *XML Schema*.

***Suppression de la hiérarchie dans les schémas* GER**

La réduction des structures *XML Schemas* en un type d'entités par type de pages trouve son origine dans un souhait d'automatiser autant que possible le processus.

Cependant, pour des types de pages très complexes, l'application de cette règle conduit à des schémas peu lisibles.

On pourrait donc envisager plutôt une approche semi-automatique lors de laquelle *DB-Main* demanderait à l'utilisateur de confirmer ou refuser chaque transformation. Bien entendu, cette solution demanderait une intervention de l'utilisateur, mais le temps consacré pourrait probablement être récupéré par la suite grâce à une meilleure lisibilité des schémas.

En conclusion, sur le plan des résultats obtenus, notre méthodologie remplit très correctement les fonctions pour lesquelles elle a été développée. Sur le plan pratique, il reste du travail pour améliorer l'efficacité des outils de support afin d'augmenter la facilité d'utilisation et la rapidité d'exécution.

Chapitre 7

Bibliographie

[Adelberg] Adelberg B. «NoDoSE—a tool for semi-automatically extracting structured and semistructured data from text documents», in *Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, Seattle, 1998, pages 283-294.

[Baumgartner et al.] Baumgartner R., Flesca S. et Gottlob G. «Visual Web Information Extraction with Lixto», in *Proceedings of 27th International Conference on Very Large Data Bases*, Rome, 2001, pages 119-128.

[Berners-Lee et al.] Berners-Lee T., Hendler J. et Lassila O. «The Semantic Web». *Scientific American*, 05/2001.

[Chung et al.] Chung C., Gertz M. et Sundaresan N. «Reverse Engineering for Web Data: From Visual to Semantic Structures», in *Proceedings of 18th International Conference on Data Engineering*, San Jose, 2002, pages 53-63.

[Cowan] Cowan J. «TagSoup - Keep On Truckin'», page d'accueil de TagSoup. Page consultée le 20/05/2004.
http://mercury.ccil.org/~cowan/XML/tagsoup/

[Crescenzi et al.] Crescenzi V., Mecca G. et Merialdo P. «Road Runner : Towards automatic data extraction from large Web sites», in *Proceedings of 27th International Conference on Very Large Data Bases*, Rome, 2001, pages 109-118.

[DB-Main] Jean-Marc Hick. «DB-Main - A R&D Project in Database Applications Engineering and CASE Technology» (04/2002), sur le site du programme DB-Main. Page consultée le 20/05/2004.
http://www.info.fundp.ac.be/~dbm/informations.html.

[Delcroix] Christine Delcroix. «Conventions de représentation d'un DTD dans le modèle GER», LIBD (document interne), 2001.

[Embley et al.] Embley D, Campbell D., Jiang Y., Liddle S., Kai Ng Y.-K., Quass D. et Smith R. «Conceptual-model-based data extraction from multiple-record web pages», *Journal of Data and Knowledge Engineering*, volume 31(3), 1999, 227-251.

[Estiévenart et al.] Estiévenart F., François A., Henrard J. et Hainaut J.-L. «A tool-supported method to extract data and schema from web sites», in *Proceedings of 5th International Workshop on Web Site Evolution*, Amsterdam, 2003, pages 3-11.

[François] Aurore François. «The XML Schema model in DB-Main». CETIC (document interne), 2002.

[Freitag] Freitag D. «Machine Learning for Information Extraction in Informal Domains», *Machine Learning*, volume 39(2-3), 2000, pages 169-202.

[Hainaut 1995] Hainaut J.-L. *Transformation-based Database Engineering*, FUNDP, 1995.

[Hainaut 2001] Jean-Luc Hainaut. *Syllabus : Ingénierie des bases de données*, FUNDP, 2001.

[Hainaut 2002] Jean-Luc Hainaut. *Introduction to Database Reverse Engineering*, LIBD Publish, 2002, 160p. (3ème Edition).

[Hainaut et al.] Hainaut J.-L., Englebert V., Henrard J., Hick J.-M. et Roland D. «Database Evolution : the DB-MAIN Approach», in *Proceedings of 13th International Conference on the Entity-Relationship Approach*, Manchester, 1994, pages 112-131.

[Laender et al. 2002] Laender A., Ribeiro-Neto B., da Silva A. et Teixeira J. «A Brief Survey of Web Data Extraction Tools», *SIGMOD Record*, volume 31(2), 2002, pages 84-93.

[Laender et al. 2002] Laender A., Ribeiro-Neto B., da Silva A. et Silva E. «Representing Web Data as Complex Objects», *Electronic Commerce and Web Technologies*, Berlin, 2000, pages 216-228.

[Liu et al.] Liu L, Pu C. et Han W. «XWRAP: An XML-enabled Wrapper Construction System for Web Information Sources», in *Proceedings of 16th International Conference on Data Engineering*, San Diego, 2000, pages 611-621.

[Marchal] Benoît Marchal. *XML by Example*. Que, 2002, 495p. (2ème édition).

[Soderland] Soderland S. «Learning information extraction rules for semi-structured and free text», *Machine Learning*, volume 34(1-3), 1999, pages 233-272.

[W3Ca] World Wide Web Consortium. «Leading the Web to its full potential», page d'accueil du World Wide Web Consortium. Page consultée le 20/05/2004.
http://www.w3c.org.

[W3Cb] W3C's HTML Working Group. «HTML 4.01 Specification» (24/12/1999), sur le site du World Wide Web Consortium. Page consultée le 20/05/2004.
http://www.w3.org/TR/html4/

[W3Cc] XML Core Working Group. «Extensible Markup Language (XML) 1.0 (Third Edition)» (04/02/2004), sur le site du World Wide Web Consortium. Page consultée le 20/05/2004.
http://www.w3.org/TR/2004/REC-xml-20040204/

[W3Cd] W3C XML Schema Working Group. «XML Schema Part 1: Structures» (02/05/2001), sur le site du World Wide Web Consortium. Page consultée le 20/05/2004.
http://www.w3.org/TR/xmlschema-1/

[W3Ce] W3C XML Schema Working Group. «XML Schema Part 2: Datatypes» (02/05/2001), sur le site du World Wide Web Consortium. Page consultée le 20/05/2004.
http://www.w3.org/TR/xmlschema-2/

[W3Cf] XSL Working Group. «XML Path Language (XPath) Version 1.0» (16/11/1999), sur le site du World Wide Web Consortium. Page consultée le 20/05/2004.
http://www.w3.org/TR/xpath

[W3Cg] XSL Working Group. «XSL Transformations (XSLT) Version 1.0» (16/11/1999), sur le site du World Wide Web Consortium. Page consultée le 20/05/2004.
http://www.w3.org/TR/xslt.

Annexe A

A.1 Code *Meta* correspondant aux figures (chapitre 3, section 3.5.1)

Figure 3.8

Code *XML* de la racine

```
<meta:htmldescription xmlns:meta="http://www.cetic.be/FR/CRAQ-DB.htm">
</meta:htmldescription>
```

Figure 3.9

Code *XML* : déclaration des éléments du type de page *Personnel*

```
<meta:htmldescription xmlns:meta="http://www.cetic.be/FR/CRAQ-DB.htm">
    <meta:element name="Personnel"></meta:element>
    <meta:element name="Identite"></meta:element>
    <meta:element name="Photo"></meta:element>
    <meta:element name="Fonction"></meta:element>
    <meta:element name="DescriptionFonction"></meta:element>
    <meta:element name="Departement"></meta:element>
    <meta:element name="Coordonnées"></meta:element>
    <meta:element name="Telephone"></meta:element>
    <meta:element name="Courriel"></meta:element>
</htmldescription>
```

Figure 3.10

Code *XML* : illustration du mécanisme de référence

```
meta:element name="Coordonnees">
    ...
    <meta:element ref="Telephone"/>
    <meta:element ref="Courriel"/>
    ...
</meta:element>
```

Figure 3.11

Code *XML* : *Coordonnees* peut contenir de 0 à 2 éléments *Telephone*.

```
meta:element name="Coordonnees">
    ...
    <meta:element ref="Telephone" min_card="0" max_card="2"/>
    <meta:element ref="Courriel"/>
    ...
</meta:element>
```

Figure 3.12

Code *HTML* : les coordonnées ont la forme d'un tableau

```
<table>
    <tr>
        <td><b>Téléphone:</b></td><td>081/990033</td>
    </tr>
    <tr>
        <td><b>Courriel :</b></td><td>animbus@company.com</td>
    </tr>
</table>
```

Code *XML* du concept *Coordonnees* dans notre formalisme

```
<meta:element name="Coordonnees">
    <table>
        <meta:element ref="Telephone" min_card="0" max_card="2"/>
        <meta:element ref="Courriel"/>
    </table>
</meta:element>
```

Figure 3.13

Code *XML* : représentation complète du concept *Telephone*

```
<meta:element name="Telephone"/>
    <tr>
        <td><b>Téléphone: </b></td> <!-- représentation d'une constante-->
        <td><meta:value/></td> <!-- représentation d'une donnée-->
    </tr>
</meta:element>
```

Figure 3.14

Code *HTML* : *DescriptionFonction* est un texte contenant des balises

```
<p>
    Le <i>responsable informatique</i> est en charge de la maintenance et du
    renouvellement du matériel informatique du département. Il est en outre
    responsable du bon fonctionnement et de la sécurité du réseau interne.
</p>
```

Code *XML* du concept *DescriptionFonction* dans notre formalisme

```
<meta:element name="DescriptionFonction">
    <p>
        <meta:value anyContent="true"/>
    </p>
</meta:element>
```

Figure 3.15

Code *HTML* positionnant la photo du membre du personnel

```
<img src="http://www.example.com/animbus.jpg"/>
```

Code *XML* du concept *Photo*

```
<meta:element name="Photo">
    <meta:value htmlattribute="src">
        <img/>
    </meta:value>
</meta:element>
```

Figure 3.16

Code *HTML* : Une grand cadre avec le nom de la société s'affiche sur les pages du type *Personnel* dans une ligne de tableau

```
<tr>
    <td class="banner">Company.com</td>
</tr>
```

Code *XML* masquant cet élément dénué d'intérêt

```
<tr>
    <meta:any/>
</tr>
```

Figure 3.17

Code *XML* décrivant le lien dans le concept *Departement*

```
<meta:element name="Departement">
    <p>
        <meta:keyref>
            <a><meta:value/></a> <!-- le nom de la personne -->
        </meta:keyref>
    </p>
</meta:element>
```

Figure 3.18

Code *XML* décrivant le concept *AncreIdentite*

```
<meta:element name="AncreIdentite">
    <meta:key>
        <a/>
    </meta:key>
</meta:element>
```

Figure 3.19

Code *XML* adapté pour le concept *Telephone*

```
<meta:element name="Telephone"
    <meta:choice>
        <meta:group>
            <tr>
                <td><b>Téléphone: </b></td>
                <td><meta:value/></td>
            </tr>
        </meta:group>
        <meta:group>
            <tr>
                <td><i>Téléphone: </i></td>
                <td><meta:value/></td>
            </tr>
        </meta:group>
    </meta:choice>
</meta:element>
```

Figure 3.20

```
<meta:htmldescription xmlns:meta='"http://www.cetic.be/FR/CRAQ-DB.htm"'>
    <meta:element name="Personnel">
        <html>
            <head>
                <meta:any/>
            </head>
            <body>
                <table>
                    <tr>
                        <meta:any/>
                    </tr>
                    <tr>
                        <td>
                            <meta:element ref="AncreIdentite"/>
                            <meta:element ref="Identite"/>
                            <meta:element ref="Photo"/>
                            <h2>Fonction</h2>
                            <meta:element ref="Fonction"/>
                            <meta:element ref="DescriptionFonction"/>
                            <h2>Département</h2>
                            <meta:element ref="Departement"/>
                            <h2>Coordonnées</h2>
                            <meta:element ref="Coordonnees"/>
                        </td>
                    </tr>
                </table>
            </body>
        </html>
    </meta:element>
    <meta:element name="AncreIdentite">
        <meta:key>
            <a/>
        </meta:key>
    </meta:element>
    <meta:element name="Identite">
        <h1>
            <meta:value dirty="true"/>
        </h1>
    </meta:element>
    <meta:element name="Photo">
        <meta:value htmlAttribute="src">
            <img/>
        </meta:value>
    </meta:element>
    <meta:element name="Fonction">
        <p>
            <meta:value/>
        </p>
    </meta:element>
    <meta:element name="DescritptionFonction">
        <p>
            <meta:value anyContent="true"/>
        </p>
    </meta:element>
    <meta:element name="Departement">
        <p>
            <meta:keyref>
                <a>
```

```
                    <meta:value/>
                </a>
            </meta:keyref>
        </p>
    </meta:element>
    <meta:element name="Coordonnees">
        <table>
            <meta:element ref="Telephone" min_card="0" max_card="2"/>
            <meta:element ref="Courriel"/>
        </table>
    </meta:element>
    <meta:element name="Telephone">
        <tr>
            <td>
                <b>Téléphone :</b>
            </td>
            <td>
                <meta:value/>
            </td>
        </tr>
    </meta:element>
    <meta:element name="Courriel">
        <tr>
            <td>
                <b>Courriel :</b>
            </td>
            <td>
                <meta:value/>
            </td>
        </tr>
    </meta:element>
</meta:htmldescription>
```

A.2 Code *XML Schema* correspondant aux figures (chapitre 3, section 3.6.1)

Figure 3.22

Code *XML Schema* du concept *Fonction*

```
<xsd:element name="Fonction" type="xsd:string"/>
```

Figure 3.23

Code *XML Schema* du concept *DescriptionFonction*

```
<xsd:element name="DescriptionFonction">
    <xsd:complexType mixed="true">
        <xsd:sequence>
            <xsd:any processContents="skip" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

Figure 3.24

Code *XML Schema* du concept *Identite*

```
<xsd:element name="Identite">
    <xsd:complexType>
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                <xsd:attribute name="dirty" type="xsd:boolean"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
</xsd:element>
```

Figure 3.25

Code *XML Schema* du concept *Coordonnees*

```
<xsd:element name="Coordonnees">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="Telephone" minOccurs="0" maxOccurs="2"/>
            <xsd:element ref="Courriel"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

Figure 3.26

Code *XML Schema* du concept *AncreIdentite*

```
<xsd:element name="AncreIdentite">
    <xsd:complexType>
        <xsd:attribute name="key" type="xsd:string"/>
    <xsd:complexType>
</xsd:element>
```

Figure 3.28

Code *XML Schema* du concept *Coordonnees* (non-ordonné)

```
<xsd:element name="Coordonnees">
    <xsd:complexType>
        <xsd:all>
            <xsd:element ref="Telephone"/>
            <xsd:element ref="Courriel"/>
        </xsd:all>
    </xsd:complexType>
</xsd:element>
```

Figure 3.29

Code *XML Schema* du concept *Coordonnees* (structure de choix)

```
<xsd:element name="Coordonnees">
    <xsd:complexType>
        <xsd:choicel>
            <xsd:element ref="Telephone"/>
            <xsd:element ref="Courriel"/>
        </xsd:choice>
    </xsd:complexType>
</xsd:element>
```

Figure 3.30

Code *XML Schema* complet du type de pages *Personnel*

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" at-
tributeFormDefault="unqualified">
    <xsd:element name="Personnel">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="AncreIdentite"/>
                <xsd:element ref="Identite"/>
                <xsd:element ref="Photo"/>
                <xsd:element ref="Fonction"/>
                <xsd:element ref="DescriptionFonction"/>
                <xsd:element ref="Departement"/>
                <xsd:element ref="Coordonnees"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="AncreIdentite">
        <xsd:complexType>
            <xsd:attribute name="key" type="xsd:anyURI"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="Identite">
        <xsd:complexType>
            <xsd:simpleContent>
                <xsd:extension base="xsd:string">
                    <xsd:attribute name="dirty" type="xsd:boolean"/>
                </xsd:extension>
            </xsd:simpleContent>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="Photo" type="xsd:string"/>
    <xsd:element name="Fonction" type="xsd:string"/>
    <xsd:element name="DescriptionFonction">
        <xsd:complexType mixed="true">
```

```
            <xsd:sequence>
                <xsd:any processContents="skip" maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="Departement">
        <xsd:complexType>
            <xsd:simpleContent>
                <xsd:extension base="xsd:string">
                    <xsd:attribute name="keyref" type="xsd:anyURI"/>
                </xsd:extension>
            </xsd:simpleContent>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="Coordonnees">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="Telephone" minOccurs="0" maxOccurs="2"/>
                <xsd:element ref="Courriel"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="Telephone" type="xsd:string"/>
    <xsd:element name="Courriel" type="xsd:string"/>
</xsd:schema>
```

Annexe B

B.1 Enrichissement sémantique du type de pages DVD réalisé à la main

B.1.1 Une première version du *Meta*

Le type de page *DVD* regroupe tous les fichiers *HTML* contenus dans le répertoire *DVDs*. Ce type de pages sera décrit dans un fichier *Meta* nommé *DVD.xml*.

Par convention, nous introduisons en tant que premier fils de l'élément racine <*htmldescription*> un élément de type <*meta:element*> dont l'attribut *name* prend comme valeur le nom du type de pages. Cet élément est le concept principal et contient la structure globale propre à tout document *HTML* :

```
<htmldescription xmlns:meta="http://www.cetic.be/FR/CRAQ-DB.htm">
    <meta:element name="DVD">
        <html>
            <head>
            </head>
            <body>
            </body>
        </html>
    </meta:element>
</htmldescritpion>
```

Pour obtenir le premier prototype, nous allons procéder à une comparaison de deux pages du dossier, à savoir *D0001.htm* (figure B.1) et *D0002.htm* (figure B.2).

Avant de nous lancer dans l'analyse de la structure visible des pages, nous allons garnir le premier fils de l'élément <*html*>, à savoir l'élément <*head*>. Pour ce faire, nous allons consulter principalement le code source, car, hormis le titre de la page, ses autres fils sont invisibles dans une fenêtre de navigateur. Voici ci-dessous le code correspondant à la section <*head*> pour les deux pages de référence :

page D0001.htm

```
<head>
    <title>Le Seigneur des Anneaux: La Communauté de l'Anneau</title>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"/>
    <link rel="stylesheet" href="../LittleZon.css" type="text/css"/>
</head>
```

page D0002.htm

```
<head>
    <title>Gladiator</title>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"/>
    <link rel="stylesheet" href="../LittleZon.css" type="text/css"/>
</head>[1]
```

Nous constatons que cet élément a trois fils et que cette structure est commune aux deux pages. La seule différence réside dans le noeud de texte interne à l'élément <*title*> qui correspond au texte affiché dans la barre de titre. Cet élément étant propre à chaque page, nous le déclarons dans le *Meta* comme concept à part entière et le nommons *TitrePage*. En voici la description dans le *Meta* :

```
<meta:element name="TitrePage">
    <title>
        <meta:value/>
    </title>
<meta:element>
```

Les deux autres fils sont des éléments classiques des pages *HTML*, à savoir une meta-description et un lien vers une feuille de style. Nous les recopions dans le *Meta* en omettant leurs attributs. Ainsi, dans le prototype, l'élément <*head*> aura la forme suivante :

```
<head>
    <meta:element ref="TitrePage"/>
    <meta/>
    <link/>
</head>
```

Figure B.1 - La page D0001.htm vue dans un navigateur

1. Lorque le balisage *HTML* des deux pages de référence est identique, nous ne reprendrons pour l'illustration que celui de la première page, *D0001.htm*

Figure B.2 - La page D0002.htm vue dans un navigateur

Pour l'analyse de la section <*body*>, nous allons consulter les pages dans un navigateur pour tenter d'identifier et hiérarchiser les concepts qui les composent, et nous consulterons ensuite le code source pour découvrir les choix de présentation de ces concepts.

D'un simple coup d'oeil à ces deux pages, nous pouvons constater que l'élément central de ce type de page est la description d'un produit de type DVD (cadre rouge dans les captures d'écran), les autres éléments (cadres verts) n'étant que des éléments de présentation (logos, étiquette du site) ou de navigation (colonne de gauche).

Si, d'un point de vue informationnel, seule la zone encadrée de rouge nous intéresse, nous devons tout de même tenir compte du reste dans le fichier *Meta* afin qu'il décrive le type de pages dans son entièreté.

Le formalisme *Meta* prévoit un mécanisme de *wild-card* pour traiter les éléments inintéressants. Pour rappel, l'élément réservé <*meta:any*> permet d'enrober le contenu intégral d'un noeud dénué d'intérêt.

Nous allons faire usage ces *wild-cards* pour cacher les zones encadrées de vert. Pour déterminer à quel endroit il convient de les placer dans le balisage *HTML* nous devons consulter le code source. Voici l'illustration pour le bandeau central :

```
<td> <!--début colonne centrale>
    <table width="100%"  border="0" cellspacing="0" cellpadding="0">
        <tr>
            <td width="100%" height="100" bgcolor="#6666CC" valign="middle"
                align="center" class="banner1">LittleZon.com</td>
        </tr>
    </table> <!--Fin bandeau-->
    <table width="100%"  border="0" cellspacing="0" cellpadding="0">
        <!-- Début ligne Craq-Reverse-->
```

```
            <tr align="left" valign="middle">
                <td nowrap class="txtbody">
                    <img src="../images/h2.gif" width="2" height="1"/>
                        <b>Projet Craq-Reverse: Une étude de cas.</b>
                </td>
            </tr>
            <tr>
                <td><img src="../images/v2.jpg" width="2" height="1"/></td>
            </tr>
            <tr>
                <td width="100%" height="1"><img src="../images/pixel6666cc.jpg"
                    width="100%" height="1"/></td>
            </tr> <!-- Fin ligne Craq Reverse-->
        </table>
        ... <!--contenu principal-->
    </td> <!fin colonne centrale>
```

Nous remarquons que l'entiereté de la colonne centrale est comprise dans un élément <*td*> qui représente une cellule de tableau. La colonne est composée de deux éléments <*table*> (contenant le bandeau de titre et la ligne « Projet RetroWeb: Une étude de cas») et du contenu principal (non détaillé dans la section de code reprise ci-dessus). Les deux éléments <*table*> doivent être cachés dans le *Meta*.

Etant donné que ces deux éléments sont frères et qu'ils ont d'autres frères (nous avons noté que leur père <*td*> englobait toute la colonne centrale) nous ne pouvons les regrouper dans une seule *wild-card*, car nous enfreindrions la règle qui interdit aux <*meta:any*> d'avoir des frères. Nous intégrons donc la structure à l'intérieur du balisage <*table*> - </*table*> pour chacun des deux éléments et nous obtenons ceci dans le fichier *Meta* :

```
<td>
    <table>
        <meta:any/>
    </table>
    <table>
        <meta:any/>
    </table>
    <!--description du contenu>
</td>
```

Nous procédons de manière similaire pour les autres zones inintéressantes et pouvons désormais nous consacrer au coeur de la page (cadre rouge).

Comme nous l'avons noté plus haut, le contenu principal de la page est constitué de l'ensemble des fils de la balise <*td*> qui constitue la colonne centrale à l'exception des deux éléments <*table*> dissimulés par les *wild-cards*.

Nous constatons immédiatement de visu que chacune des deux pages contient trois niveaux de titres :

- niveau 1 : le titre du film
- niveau 2 : la description du film, le mode de commande et l'avis des visiteurs
- niveau 3 : le synopsis, les détails techniques et les réactions.

Procédons maintenant à l'analyse en détail de ces différents éléments afin de leur réserver un traitement adéquat.

• Le titre du film

Sans surprise, nous constatons que la valeur de cet élément est propre à chaque page. Nous en déduisons qu'il s'agit d'une donnée importante qui devra être extraite et donc représentée dans le fichier *Meta* par un élément <*meta:value/*>.

Nous remarquons également que sa valeur est identique à celle du premier fils de l'élément

<*head*> décrit plus haut. Cependant, le formalisme *Meta* interdit dans ce cas de réutiliser le même <*meta:element*> afin de garantir la cohérence des donnés. En effet, rien ne peut garantir a-priori que la correspondance des valeurs sera vérifiée dans toutes les pages. Si la redondance se confirme, nous nous chargerons de l'éliminer lors de la phase de conceptualisation.

Nous créons donc un nouvel élément global et nous nous référons au code source afin d'obtenir les éléments de présentation :

Code *HTML* source
```
<h1>Le Seigneur des Anneaux: La Communauté de l'Anneau</h1>
```

Représentation *Meta*
```
<meta:element name="TitreFilm">
    <h1>
        <meta:value/>
    </h1>
</meta:element>
```

• La description du film

La description du film apparaît immédiatement comme étant la partie la plus complexe du contenu central. Elle est divisée en cinq concepts :

- un titre,
- une image,
- un synopsis,
- des détails,
- des réactions.

• Le titre

En comparant les deux pages, nous remarquons que ce titre est composé d'éléments communs (mention "Le film (Ref: " et la parenthèse fermante) et une valeur propre à chaque page qui représente la référence interne du produit et qui correspond au nom de fichier sans l'extension *.htm*.

Le formalisme *Meta* ne permettant pas à un élément <*meta:value*/> d'avoir des frères de type texte, nous allons marquer le titre entier comme une valeur significative. Et afin de noter que la donnée devra être amputée des éléments communs, nous donnons la valeur *true* à l'attribut *dirty* de l'élément <*meta:value*/>. La consultation du code source nous indique les choix de présentation :

Code *HTML* source
```
<h2>Le film (Ref: D0001)</h2>
```

Reste à donner un nom représentatif à ce concept et à l'intégrer au *Meta* :

Représentation *Meta*
```
<meta:element name="Reference">
    <h2>
        <meta:value dirty="true"/>
    </h2>
</meta:element>
```

• L'image

L'image est spécifique à chaque page, nous devons donc la déclarer comme donnée pertinente et récupérer l'adresse du fichier source grâce à l'utilisation de l'attribut *HTMLAttribute* dans l'élément <*meta:value*>. Ainsi, le code source

```
<img class="imgprinc" src="../images/imgD0001.jpg" align="right" alt="affiche"/>
```

sera représenté comme suit dans le fichier *Meta* :

```
<meta:element name="Image">
    <meta:value HTMLAttribute="src">
        <img/>
    </meta:value>
</meta:element>
```

- **Le synopsis**

Dans les deux pages de référence, le synopsis a le même intitulé. Il s'agit donc apparemment d'une constante textuelle et non d'une valeur à extraire. Par conséquent, nous l'intégrons tel quel dans la description *Meta*.

Le texte sous l'intitulé est, par contre, propre à chaque page. Nous allons donc le marquer comme donnée à extraire. Reste à consulter le code source pour découvrir le balisage du concept.

```
<img class="imgprinc" src="../images/imgD0001.jpg"> <!--élément précédent>
<h3>Synopsis</h3>
<p>Dans ce premier chapitre de la trilogie, le jeune et timide Hobbit, <i>Frodon Sacquet</i>, hérite d'un anneau. Bien loin d'être une simple babiole, il s'agit de l'Anneau Unique, un instrument de pouvoir absolu qui permettrait à Sauron, le Seigneur des ténèbres, de régner sur la Terre du Milieu et de réduire en esclavage ses peuples. À moins que <i>Frodon</i>, aidé d'une Compagnie constituée de Hobbits, d'Hommes, d'un Magicien, d'un Nain, et d'un Elfe, ne parvienne à emporter l'Anneau à travers la Terre du Milieu jusqu'à la <i>Crevasse du Destin</i>, lieu où il a été forgé, et à le détruire pour toujours...<br />
    Un tel périple signifie s'aventurer très loin en <i>Mordor</i>, les terres du Seigneur des ténèbres, où est rassemblée son armée d'Orques maléfiques... La Compagnie doit non seulement combattre les forces extérieures du mal mais aussi les dissensions internes et l'influence corruptrice qu'exerce l'Anneau lui-même. L'issue de l'histoire à venir est intimement liée au sort de la Compagnie.</p>
```

Nous remarquons que la donnée à extraire est composée de plusieurs noeuds (texte et balises). En effet, certaines parties du texte sont formatées en italique. Dès lors, nous associons à l'élément *<meta:value/>* l'attribut *anyContent* mis à la valeur *true*, ce qui permet de représenter une donnée composée de texte et de balises *HTML*. Nous obtenons dès lors la description suivante pour le concept *Synopsis* :

```
<meta:element name="Synopsis">
    <h3>
        Synopsis
    </h3>
    <p>
        <meta:value anyContent="true"/>
    </p>
</meta:element>
```

- **Les détails**

Le libelle des détails mène à la même remarque que pour le concept *Synopsis*.

Ensuite, nous identifions 5 sous-éléments (*Realisateur*, *Annee*, *Acteurs*, *Genre* et *Duree*) qui présentent la même structure, à savoir un libellé commun aux deux pages mis en gras et une donnée spécifique à chaque page. A titre d'exemple, voici le code source suivi de la structure de l'élément *Genre* dans le fichier *Meta* :

Code *HTML* source

```
<p>
    <b>Genre: </b>
     Fantastique
</p>
```

Représentation *Meta*

```
<meta:element name="Genre">
    <p>
        <b>
            Genre:
        </b>
        <meta:value/>
    </p>
</meta:element>
```

En définissant selon le même schéma chacun des sous-élément, nous obtenons la description suivante pour le concept *Details* :

```
<meta:element name="Details">
    <h3>
        Détails
    </h3>
    <meta:element ref="Realisateur"/>
    <meta:element ref="Annee"/>
    <meta:element ref="Acteurs"/>
    <meta:element ref="Genre"/>
    <meta:element ref="Duree"/>
</meta:element>
```

Notons toutefois que les données à extraire pour les éléments *Realisateur* et *Acteurs* sont complexes. En effet, le noeud de texte variable de l'élément *Realisateur* contient un prénom et un nom (donnée composée), tandis que celui de l'élément *Acteurs* contient une suite de couples nom-prénom (donnée multivaluée et composée). Par conséquent, nous marquons les <*meta:value*> de ces éléments avec l'attribut *dirty* mis à la valeur *true* pour prévoir un nettoyage de ces données après la phase d'extraction.

- **Les réactions**

Le libellé est encore une fois identique aux deux pages.

Nous remarquons pour les réactions une différence entre les deux pages. En effet, la première page comporte trois paragraphes et la seconde en contient seulement deux. Chacun de ces paragraphes contient des informations de même sémantique. Le concept *Reaction* est donc multivalué.

Etant donné que le libelle "Réactions" n'est présent qu'une seule fois, nous ne pouvons l'intégrer dans la description du concept; il sera donc repris directement dans celle de son concept père, à savoir le concept principal.

Ajoutons qu'un élément *Reaction* est composé de deux valeurs (le nom de l'auteur de la critique et le texte de celle-ci) qu'il nous paraît intéressant de séparer dans l'optique de la création d'une base de données. Etant donné que les spécifications du formalisme *Meta* interdisent formellement la déclaration de concepts contenant à la fois un élément <*meta:value/*> et une ou plusieurs références à d'autres concepts (<*meta:element ref=""*>) nous définissons deux sous-concepts au sein de *Reaction* que nous nommons *Identite* et *Critique*. Après un coup d'oeil au code source, nous en arrivons à la définition des trois éléments suivants :

Code *HTML* source

```
<p>
    <b>Première:</b>
    Le résultat, enthousiasmant au-delà de ce qu'on pouvait espérer, frôle l'idéal
    de <i>cinéma pur</i> que très) peu (Disney, Hitchcock, Powell, Kurosawa) ont réussi
    à atteindre.
</p>
```

Description *Meta*

```
<meta:element name="Identite">
    <b>
        <meta:value/>
    </b>
</meta:element>

<meta:element name="Critique">
    <meta:value anyContent="true"/>
</meta:element>

<meta:element name="Reaction">
    <p>
        <meta:element ref="Identite"/>
```

```
        <meta:element ref="Critique"/>
    </p>
</meta:element>
```

• Le mode de commande

La partie de la page décrivant le mode de commande est totalement commune aux deux pages. Elle est consituée d'un titre et d'une phrase dans laquelle est inséré un lien vers une boîte de messagerie électronique. En voici le code source :

```
<h2>Commander</h2>
<p>
    Pour commander, envoyez un
    <a href="mailto:order@littlezon.com">mail</a>
    en mentionnant la référence reprise à côté du titre du film.
</p>
```

Ces éléments ne nous apparaissent pas comme des données fondamentales du domaine d'application, aussi les reprenons-nous tels quels au sein du concept principal dans la description *Meta*.

• L'avis des visiteurs

La partie concernant l'avis des visiteurs est pratiquement commune aux deux pages. Elle est composée de deux paragraphes.

Le premier contient un texte commun et une donnée chiffrée variable en caractères gras. Celle-ci représente la cote moyenne attribuée au produit par les visiteurs du site. Le fait que cette donnée soit formatée en gras nous permettra de l'isoler par rapport au reste du paragraphe.

Le second paragraphe a une structure identique à celle du paragraphe *Commander*, il subira donc un traitement identique au sein du concept *Avis*.

Consultons maintenant le code source pour prendre connaissance du balisage de la section :

Code source *HTML*

```
<h2>L'avis des visiteurs du site</h2>
<p>
    De nombreux visiteurs du site LittleZon ont donné leur opinion sur ce film.
    Actuellement, la cote moyenne obtenue par ce titre est de
     <b>8/10</b>.
</p>
<p>
    Pour donner votre avis, envoyez-nous un
    <a href="mailto:opinion@littlezon.com">mail</a>
    précisant le numéro de référence du DVD et votre cote sur 10. Vous pouvez y
     joindre un texte décrivant votre opinion, elle sera peut-être publiée sur le site.
</p>
```

Etant donné que seule la donnée chiffrée nous intéresse dans cette section, il n'est pas nécessaire de définir un sous-concept à l'élément *Avis*. L'insertion d'un élément *<meta:value>* entre les balises de formatage ** et ** suffit. Ainsi nous obtenons dans le *Meta* :

```
<meta:element name="Avis">
    <h2>
        L'avis des visiteurs du site
    </h2>
    <p>
        De nombreux visiteurs du site LittleZon ont donné leur opinion sur ce film.
        Actuellement, la cote moyenne obtenue par ce titre est de
        <b>
            <meta:value/>
        </b>.
    </p>
    <p>
        Pour donner votre avis, envoyez-nous un
        <a href="mailto:opinion@littlezon.com">mail</a>
        précisant le numéro de référence du DVD et votre cote sur 10. Vous pouvez
```

```
            y joindre un texte décrivant votre opinion, elle sera peut-être publiée sur le
            site.
        </p>
    </meta:element>
```

A ce stade, nous avons construit les descriptions de tous les éléments du contenu principal. Reste à les intégrer dans la déclaration du concept principal :

```
<meta:element name="DVD">
    <html>
        <head>
            <meta:element ref="TitrePage"/>
            <meta/>
            <link/>
        </head>
        <body>
            <table>
                <tr>
                    <td>
                        <meta:any/>
                        <!--colonne de gauche-->
                    </td>
                    <td>
                        <table>
                            <meta:any/>
                            <!--bandeau titre-->
                        </table>
                        <table>
                            <meta:any/>
                            <!--ligne Craq-Reverse-->
                        </table>
                        <meta:element ref="TitreFilm"/>
                        <meta:element ref="Reference"/>
                        <meta:element ref="Image"/>
                        <meta:element ref="Synopsis"/>
                        <meta:element ref="Details"/>
                        <h3>Réactions</h3>
                        <meta:element ref="Reaction" max_card="3"/>
                        <h2>Commander</h2>
                        <p>
                            Pour commander, envoyez un
                            <a href="mailto:order@littlezon.com">mail</a>
                            en mentionnant la référence reprise à côté du titre du film.
                        </p>
                        <meta:element ref="Avis"/>
                    </td>
                    <td>
                        <meta:any/>
                        <!--colonne de droite-->
                    </td>
                </tr>
                <tr>
                    <meta:any/>
                    <!--fine ligne en-bas-->
                </tr>
            </table>
        </body>
    </html>
</meta:element>
```

Ainsi se termine la première version du fichier de description du type de page *DVD*. Elle sera

éventuellement complétée et affinée au fur et à mesure du processus de validation.

B.1.2 Validation des pages et affinage du *Meta*

L'outil de validation génère un journal dans la console de travail. C'est à l'intérieur de ces messages que nous découvrirons où se situent les éventuelles inconsistances par rapport au prototype.

Sans surprise, la confrontation du fichier meta aux pages *D0001.htm* et *D0002.htm* ne pose pas de problème, vu que c'est l'analyse de ces deux pages qui lui a donné naissance.

Par contre, la validation de la page *D0003.htm* se termine par le message «D0003.htm n'est pas conforme au meta». Un affinage s'impose donc.

- **Deuxième version du meta**

  Message de la console de travail :

```
HTML Father : b
HTML Position : 1
META : #text
Réalisateur:
Réalisateurs:
```

Ce message indique que l'erreur se situe dans une constante de texte. L'outil attendait la chaîne de caractères «Réalisateur» et a rencontré «Réalisateurs».

Afin de régler ce problème, il suffit de représenter le concept incriminé dans une structure de choix proposant le singulier de la constante dans la première branche et le pluriel dans la seconde :

```
<meta:element name="Realisateur">
    <meta:choice>
        <meta:group>
            <p>
                <b>Réalisateur: </b>
                <meta:value dirty="true"/>
            </p>
        </meta:group>
        <meta:group>
            <p>
                <b>Réalisateurs: </b>
                <meta:value dirty="true"/>
            </p>
        </meta:group>
    </meta:choice>
</meta:element>
```

Techniquement parlant, il eut été possible de remplacer la constante textuelle par un élément *<meta:any/>* afin d'éviter une structure de choix qui est relativement lourde. Cependant, en agissant de la sorte, n'importe quel élément ayant la même structure (un libellé en gras suivi d'une donnée, le tout au sein d'un paragraphe) pourrait être validé en tant qu'élément *Realisateur*, ce qui porterait préjudice à la cohérence sémantique des données. Dans notre cas, tout instance de type *Acteur*, *Annee*, *Genre* ou *Duree* serait accepté en tant qu'élement *Realisateur*.

Nous substituons la nouvelle description à l'ancienne et obtenons ainsi un nouveau prototype du fichier *Meta*.

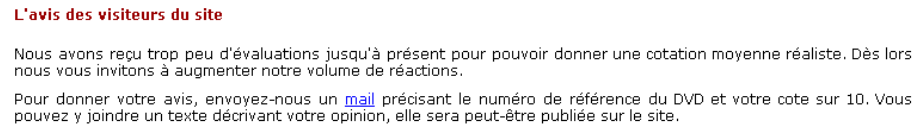
Nous reprenons le processus de validation avec le nouveau prototype. Le problème suivant survient dans la page *D0005.htm*.

- **Troisième version du meta**

A la lecture du journal, il apparaît immédiatement que le problème rencontré est de même nature que le précédent, à savoir une différence dans une constante de type texte, mais ici au sein de l'élément *Avis* :

```
HTML Father : p
HTML Position : 0
META : #text
De nombreux visiteurs du site LittleZon ont donné leur opinion sur ce film.
Actuellement, la cote moyenne obtenue par ce titre est de
Nous avons reçu trop peu d'évaluations jusqu'à présent pour pouvoir
 donner une cotation moyenne réaliste. Dès lors nous vous invitons à
 augmenter notre volume de réactions.
```

La section incriminée présentant plusieurs constantes de texte, nous décidons de consulter la page *D0005.htm* dans un navigateur afin de régler d'éventuelles autres différences de même type.


L'avis des visiteurs du site

Nous avons reçu trop peu d'évaluations jusqu'à présent pour pouvoir donner une cotation moyenne réaliste. Dès lors nous vous invitons à augmenter notre volume de réactions.

Pour donner votre avis, envoyez-nous un mail précisant le numéro de référence du DVD et votre cote sur 10. Vous pouvez y joindre un texte décrivant votre opinion, elle sera peut-être publiée sur le site.

Figure B.3 - Section *Avis* de la page *D0005.htm*

De visu, nous remarquons que seul le premier paragraphe est différent de notre description et qu'il ne contient plus de donnée à extraire.

La même méthode que pour le cas précédent s'applique ici. Nous réécrivons donc la description de l'élément *Avis* en prévoyant l'alternative :

```
<meta:element name="Avis">
    <meta:choice>
        <meta:group>
            <h2>L'avis des visiteurs du site</h2>
            <p>
                De nombreux visiteurs du site LittleZon ont donné leur opinion
                sur ce film. Actuellement, la cote moyenne obtenue par ce titre
                est de
                <b><meta:value/></b>
            </p>
            <p>...  <!-- le second paragraphe demeure identique--></p>
        </meta:group>
        <meta:group>
            <p>
                Nous avons reçu trop peu d'évaluations jusqu'à présent pour pou
                voir donner une cotation moyenne réaliste. Dès lors nous vous
                invitons à augmenter notre volume de réactions.
            </p>
            <p>...  <!-- le second paragraphe demeure identique--></p>
        </meta:group>
    </meta:choice>
</meta:element>
```

Nous reprenons le processus de validation avec le nouveau prototype. Le problème suivant survient dans le fichier *D0008.htm*.

- **Quatrième version du meta**

A nouveau l'exception se passe dans un noeud de texte constant. L'outil de validation rencontre la chaine de caractères «Voix» au lieu de «Acteurs» dans l'élément *Acteurs*.

A première vue, il suffirait d'appliquer la méthode utilisée dans les deux cas précédents et le

problème serait réglé. Cependant, en y regardant de plus près, cette différence revêt une plus grande importance que les deux autres d'un point de vue sémantique. En effet, elle indique que des comédiens participent à un film soit en tant qu'acteurs soit en tant que voix pour des personnages animés. Nous allons donc introduire un nouvel élément *Voix* qui sera référencé de manière facultative après *Acteurs* dans le concept *Details*. La référence au concept *Acteurs* devient elle aussi facultative :

```
<meta:element name="Voix">
    <p>
        <b>Voix: </b>
        <meta:value dirty="true"/>
    </p>
</meta:element>
```

La nouvelle version du *Meta* valide l'ensemble des pages du type, nous en faisons donc notre version définitive. En voici le code complet.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<meta:htmldescription xmlns:meta="http://www.cetic.be/FR/CRAQ-DB.htm">
    <meta:element name="DVD">
        <html>
            <head>
                <meta:element ref="TitrePage"/>
                <meta/>
                <link/>
            </head>
            <body>
                <table>
                    <tr>
                        <td>
                            <meta:any/>
                            <!--colonne de gauche-->
                        </td>
                        <td>
                            <table>
                                    <meta:any/>
                                    <!--bandeau titre-->
                            </table>
                            <table>
                                    <meta:any/>
                                    <!--ligne Craq-Reverse-->
                            </table>
                            <meta:element ref="TitreFilm"/>
                            <meta:element ref="Reference"/>
                            <meta:element ref="Image"/>
                            <meta:element ref="Synopsis"/>
                            <meta:element ref="Details"/>
                            <h3>Réactions</h3>
                            <meta:element ref="Reaction" max_card="3"/>
                            <h2>Commander</h2>
                            <p>
                                    Pour commander, envoyez un
                                    <a href="mailto:order@littlezon.com">mail</a>
                                    en mentionnant la référence reprise à côté du titre du film.
                            </p>
                            <meta:element ref="Avis"/>
                        </td>
                        <td>
                            <meta:any/>
```

```
                    <!--colonne de droite-->
                </td>
            </tr>
            <tr>
                <meta:any/>
                <!--fine ligne en-bas-->
            </tr>
        </table>
    </body>
</html>
</meta:element>
<meta:element name="TitrePage">
    <title>
        <meta:value/>
    </title>
</meta:element>
<meta:element name="TitreFilm">
    <h1>
        <meta:value/>
    </h1>
</meta:element>
<meta:element name="Reference">
    <h2>
        <meta:value dirty="true"/>
    </h2>
</meta:element>
<meta:element name="Image">
    <meta:value htmlAttribute="src">
        <img/>
    </meta:value>
</meta:element>
<meta:element name="Synopsis">
    <h3>Synopsis</h3>
    <p>
        <meta:value anyContent="true"/>
    </p>
</meta:element>
<meta:element name="Details">
    <h3>Détails</h3>
    <meta:element ref="Realisateur"/>
    <meta:element ref="Annee"/>
    <meta:element ref="Acteurs" min_card="0"/>
    <meta:element ref="Voix" min_card="0"/>
    <meta:element ref="Genre"/>
    <meta:element ref="Duree"/>
</meta:element>
<meta:element name="Realisateur">
    <meta:choice>
        <meta:group>
            <p>
                <b>
                    Réalisateur:
                </b>
                <meta:value dirty="true"/>
            </p>
        </meta:group>
        <meta:group>
            <p>
                <b>
```

```
                    Réalisateurs:
                </b>
                <meta:value dirty="true"/>
            </p>
        </meta:group>
    </meta:choice>
</meta:element>
<meta:element name="Annee">
    <p>
        <b>Année: </b>
        <meta:value/>
    </p>
</meta:element>
<meta:element name="Acteurs">
    <p>
        <b>Acteurs: </b>
        <meta:value dirty="true"/>
    </p>
</meta:element>
<meta:element name="Voix">
    <p>
        <b>Voix: </b>
        <meta:value dirty="true"/>
    </p>
</meta:element>
<meta:element name="Genre">
    <p>
        <b>Genre: </b>
        <meta:value/>
    </p>
</meta:element>
<meta:element name="Duree">
    <p>
        <b>Durée: </b>
        <meta:value/>
    </p>
</meta:element>
<meta:element name="Reaction">
    <p>
        <meta:element ref="Identite"/>
        <meta:element ref="Critique"/>
    </p>
</meta:element>
<meta:element name="Identite">
    <b>
        <meta:value/>
    </b>
</meta:element>
<meta:element name="Critique">
    <meta:value anyContent="true"/>
</meta:element>
<meta:element name="Avis">
    <meta:choice>
        <meta:group>
            <h2>L'avis des visiteurs du site</h2>
            <p>
                De nombreux visiteurs du site LittleZon ont donné leur opinion sur ce film. Actuelle-
ment, la cote moyenne obtenue par ce titre est de
                <b>
```

```
                        <meta:value/>
                    </b>
                    .
                    </p>
                <p>
                        Pour donner votre avis, envoyez-nous un
                        <a href="mailto:opinion@littlezon.com">mail</a>
                        précisant le numéro de référence du DVD et votre cote sur 10. Vous pouvez y
joindre un texte décrivant votre opinion, elle sera peut-être publiée sur le site.
                    </p>
            </meta:group>
            <meta:group>
                <h2>L'avis des visiteurs du site</h2>
                <p>
                    Nous avons reçu trop peu d'évaluations jusqu'à présent pour pouvoir donner une co-
tation moyenne réaliste. Dès lors nous vous invitons à augmenter notre volume de réactions.
                </p>
                <p>
                    Pour donner votre avis, envoyez-nous un
                    <a href="mailto:opinion@littlezon.com">mail</a>
                    précisant le numéro de référence du DVD et votre cote sur 10. Vous pouvez y joindre
un texte décrivant votre opinion, elle sera peut-être publiée sur le site.
                </p>
            </meta:group>
        </meta:choice>
    </meta:element>
</meta:htmldescription>
```

Annexe C

C.1 Fichier «Nouveautes.htm» généré par *Retrozilla*

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
 <metaelement name="TitrePage">
 <title><metavalue>LittleZon: Les nouveaux livres</metavalue></title>
 </metaelement>
 <meta content="text/html; charset=iso-8859-1"
http-equiv="Content-Type">
 <link type="text/css" href="../LittleZon.css" rel="stylesheet">
 <meta name="pagetype" content="Nouveaute">
</head>
<body>
<!-- Début de la table principale-->
<table cellspacing="0" cellpadding="0" border="0" width="100%">
 <tbody>
  <tr>
<!--Début colonne de gauche--> <td bgcolor="#6666cc" valign="top"
width="110"
style="background: red none repeat scroll 0%; -moz-background-clip: initial; -moz-background-origin:
initial; -moz-background-inline-policy: initial;"><metaany>
      <table cellspacing="0" cellpadding="0" border="0" width="100%">
       <tbody>
        <tr>
         <td bgcolor="#ffffff" align="center" colspan="4"
height="100" width="110"><a href="http://www.fundp.ac.be"> <img
border="0" alt="logo fundp" height="100" width="110"
src="../images/sigle_vierge_haut.jpg"></a></td>
        </tr>
        <tr>
         <td align="center" colspan="4" height="25" width="110"><a
href="http://www.fundp.ac.be"> <img border="0" alt="logo suite"
height="25" width="110" src="../images/sigle_vierge_bas6666cc.jpg"></a></td>
        </tr>
       </tbody>
      </table>
      <hr color="#ccccff" size="3" noshade="noshade">
      <hr color="#ccccff" size="1" noshade="noshade">
      <p style="margin-left: 0px;" class="menugauche"> <a
class="menugauche" href="../index.htm"><b>Accueil</b></a><br>
      </p>
      <hr color="#ccccff" size="1" noshade="noshade">
```

```
        <hr color="#ccccff" size="3" noshade="noshade">
        <p style="font-size: 6px;"> </p>
        <hr color="#ccccff" size="3" noshade="noshade">
        <hr color="#ccccff" size="1" noshade="noshade">
        <p class="menugaucheprinc">Nos produits </p>
        <hr color="#ccccff" size="1" noshade="noshade">
        <p class="menugauche"><b>Livres</b><br>
        <img height="10" width="12" src="../images/balle_orange.gif"> <a
     style="color: rgb(255, 153, 0);" class="menugauche"
     href="NewLivres.htm">Nouveaut&eacute;s</a><br>
        <img height="10" width="12" src="../images/balle_blanche.gif"> <a
     class="menugauche" href="../Listes/ListLivres.htm">Liste
    compl&egrave;te</a> </p>
        <hr color="#ccccff" size="1" noshade="noshade">
        <hr color="#ccccff" size="1" noshade="noshade">
        <p class="menugauche"><b>CDs</b><br>
        <img height="10" width="12" src="../images/balle_blanche.gif"> <a
     class="menugauche" href="NewCDs.htm">Nouveaut&eacute;s</a><br>
        <img height="10" width="12" src="../images/balle_blanche.gif"> <a
     class="menugauche" href="../Listes/ListCDs.htm">Liste compl&egrave;te</a>
        </p>
        <hr color="#ccccff" size="1" noshade="noshade">
        <hr color="#ccccff" size="1" noshade="noshade">
        <p class="menugauche"><b>DVDs</b><br>
        <img height="10" width="12" src="../images/balle_blanche.gif"> <a
     class="menugauche" href="NewDVDs.htm">Nouveaut&eacute;s</a><br>
        <img height="10" width="12" src="../images/balle_blanche.gif"> <a
     class="menugauche" href="../Listes/ListDVDs.htm">Liste compl&egrave;te</a>
        </p>
        <hr color="#ccccff" size="1" noshade="noshade">
        <hr color="#ccccff" size="3" noshade="noshade"><!-- Fin colonne de gauche--> </metaany></td>
        <td valign="top" width="100%"><!-- Début bandeau du haut-->
        <table cellpadding="0" cellspacing="0" border="0" width="100%"
     style="background: red none repeat scroll 0%; -moz-background-clip: initial; -moz-background-origin:
initial; -moz-background-inline-policy: initial;">
          <metaany> <tbody>
            <tr>
              <td class="banner1" align="center" valign="middle"
     bgcolor="#6666cc" height="100" width="100%">LittleZon.com</td>
            </tr>
          </tbody></metaany>
        </table>
    <!--Fin bandeau-->
        <table cellpadding="0" cellspacing="0" border="0" width="100%"
     style="background: red none repeat scroll 0%; -moz-background-clip: initial; -moz-background-origin:
initial; -moz-background-inline-policy: initial;">
          <metaany><!-- Début ligne Craq-Reverse--> <tbody>
            <tr valign="middle" align="left">
              <td class="txtbody" nowrap="nowrap"> <img height="1"
     width="2" src="../images/h2.gif"> <b>Projet RetroWeb: Une
    &eacute;tude de cas.</b> </td>
            </tr>
            <tr>
              <td><img height="1" width="2" src="../images/v2.jpg"></td>
            </tr>
            <tr>
              <td height="1" width="100%"><img height="1" width="100%"
     src="../images/pixel6666cc.jpg"></td>
            </tr>
```

```
<!-- Fin ligne Craq Reverse--><!--Début texte principal-->
      </tbody></metaany>
    </table>
    <metaelement name="TitreContenu">
    <h1
 style="background: yellow none repeat scroll 0% 50%; -moz-background-clip: initial; -moz-background-
origin: initial; -moz-background-inline-policy: initial; margin-bottom: 0px; margin-left: 11px; color:
green;"><metavalue>LittleBookZon:
 les nouveaut&eacute;s</metavalue></h1>
    </metaelement>
    <table width="100%" cellpadding="0" class="tablenews">
     <tbody>
      <tr>
       <metaelement name="Produit"><td
 style="background: yellow none repeat scroll 0%; -moz-background-clip: initial; -moz-background-origin:
initial; -moz-background-inline-policy: initial;"><metaelement
 name="NomProduit">
       <h2
 style="background: yellow none repeat scroll 0% 50%; -moz-background-clip: initial; -moz-background-
origin: initial; -moz-background-inline-policy: initial; color: green;"><metavalue>Harry
 Potter et l'Ordre du Phoenix</metavalue></h2>
       </metaelement> <a href="../Livres/L0010.htm"><metaelement
 name="Image"><metavalue htmlattribute="src"><img alt="couverture"
 align="left" src="../images/smallimgL0010.jpg" class="imgnews"
 style="background: yellow none repeat scroll 0%; -moz-background-clip: initial; -moz-background-origin:
initial; -moz-background-inline-policy: initial;"></metavalue></metaelement></a>
       <metaelement name="Auteur">
       <h3
 style="background: yellow none repeat scroll 0%; -moz-background-clip: initial; -moz-background-origin:
initial; -moz-background-inline-policy: initial;">Auteur</h3>
       <p
 style="background: yellow none repeat scroll 0% 50%; -moz-background-clip: initial; -moz-background-
origin: initial; -moz-background-inline-policy: initial; color: green;"><metavalue>
 Joanne-K. Rowling </metavalue></p>
       </metaelement><metaelement name="Description">
       <h3
 style="background: yellow none repeat scroll 0%; -moz-background-clip: initial; -moz-background-origin:
initial; -moz-background-inline-policy: initial;">En
 bref...</h3>
       <p
 style="background: yellow none repeat scroll 0% 50%; -moz-background-clip: initial; -moz-background-
origin: initial; -moz-background-inline-policy: initial; color: green;"><metavalue>
 Probablement une des sorties les plus attendues de l'histoire de la
 litt&eacute;rature... voil&agrave; enfin en version fran&ccedil;aise le
 5&egrave;me volume des aventures du plus c&eacute;l&egrave;bre des
 apprentis sorciers. Plus de 900 pages de plaisir et de suspense qui
 vous feront d&eacute;couvrir les premiers pas de <i>Harry</i> dans
 l'adolescence. Ca valait la peine d'attendre. </metavalue></p>
       </metaelement>
       <h3>Pour en savoir plus...</h3>
       <p
 style="background: red none repeat scroll 0%; -moz-background-clip: initial; -moz-background-origin:
initial; -moz-background-inline-policy: initial;"><metaany>
 D&eacute;couvrez toutes les informations sur ce livre
 (r&eacute;sum&eacute;, critiques, d&eacute;tails,...) en vous reportant
 &agrave; la <a href="../Livres/L0010.htm">page</a> qui y est
 consacr&eacute;e. </metaany></p>
       <metaelement name="NumeroSerie" min_card="0">
       <h3 class="relief"
```

```
     style="background: yellow none repeat scroll 0% 50%; -moz-background-clip: initial; -moz-background-
origin: initial; -moz-background-inline-policy: initial; color: green;"><metavalue>Num&eacute;ro
    dans la s&eacute;rie : 5</metavalue></h3>
              </metaelement> </td>
              </metaelement></tr>
             <tr>
              <td>
              <h2>Ant&eacute;christa</h2>
              <a href="../Livres/L0011.htm"><img alt="couverture"
     align="left" src="../images/smallimgL0011.jpg" class="imgnews"></a>
              <h3>Auteur</h3>
              <p> Am&eacute;lie Nothomb </p>
              <h3>En bref...</h3>
              <p> Un grand cri de souffrance et d'espoir pour les
    adolescents en qu&ecirc;te de reconnaissance et d'une identit&eacute;
    propre.<br>
    Un hymne &agrave; la libert&eacute; face &agrave; la tyranie et la
    dictature des dominateurs quels qu'ils soient; la vie et rien d'autre
    la revanche de l'opprim&eacute; sur l'oppresseur . </p>
              <h3>Pour en savoir plus...</h3>
              <p> D&eacute;couvrez toutes les informations sur ce livre
    (r&eacute;sum&eacute;, critiques, d&eacute;tails,...) en vous reportant
    &agrave; la <a href="../Livres/L0011.htm">page</a> qui y est
    consacr&eacute;e.</p>
              </td>
             </tr>
            </tbody>
           </table>
           </td>
    <!--Début colonne de droite--><!--<td width="110" valign="top" bgcolor="#6666cc">
                  <table width="100%" border="0" cellpadding="0" cellspacing="0">
                      <tr>
                          <td width="110" height="25" colspan="4" align="center"><a href="http://www.ce-
tic.be">
                                  <img src="../images/cetic.gif" width="110" height="170" alt="logo_cetic" sty-
le="border-left-color:#6666CC; border-left-width:thin" " border="0"></a></td>
                      </tr>
                  </table>
                  <hr noshade size="3" color="#CCCCFF"><hr noshade size="1" color="#CCCCFF">
              </td>--> </tr>
          <tr>
            <td colspan="3" width="100%"><img height="1" width="100%"
     src="../images/pixelccccff.jpg"></td>
           </tr>
         </tbody>
    </table>
    </body>
    </html>
```

C.2 Fichier *Meta* du type de pages *Nouveautes* (version 1)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<meta:htmldescription xmlns:meta="http://www.cetic.be/FR/CRAQ-DB.htm">
    <meta:element name="Nouveautes">
        <html>
            <head>
                <meta:element ref="TitrePage"/>
                <meta/>
                <link/>
            </head>
            <body>
                <table>
                    <tr>
                        <td>
                            <meta:any/>
                        </td>
                        <td>
                            <table>
                                    <meta:any/>
                            </table>
                            <table>
                                    <meta:any/>
                            </table>
                            <meta:element ref="TitreContenu"/>
                            <table>
                                    <meta:element ref="Produit" max_card="N"/>
                            </table>
                        </td>
                    </tr>
                    <tr>
                        <meta:any/>
                    </tr>
                </table>
            </body>
        </html>
    </meta:element>
    <meta:element name="TitrePage">
        <title>
            <meta:value/>
        </title>
    </meta:element>
    <meta:element name="TitreContenu">
        <h1>
            <meta:value/>
        </h1>
    </meta:element>
    <meta:element name="Produit">
        <tr>
            <td>
                <meta:element ref="NomProduit"/>
                <meta:element ref="Image"/>
                <meta:element ref="Auteur"/>
                <meta:element ref="Description"/>
                <h3>Pour en savoir plus...</h3>
                <p>
                    <meta:any/>
                </p>
```

```
                <meta:element ref="NumeroSerie" min_card="0"/>
            </td>
        </tr>
    </meta:element>
    <meta:element name="NomProduit">
        <h2>
            <meta:value/>
        </h2>
    </meta:element>
    <meta:element name="Image">
        <meta:keyref>
            <a>
                <meta:value htmlAttribute="src">
                    <img/>
                </meta:value>
            </a>
        </meta:keyref>
    </meta:element>
    <meta:element name="Auteur">
        <h3>Auteur</h3>
        <p>
            <meta:value dirty="true"/>
        </p>
    </meta:element>
    <meta:element name="Description">
        <h3>En bref...</h3>
        <p>
            <meta:value anyContent="true"/>
        </p>
    </meta:element>
    <meta:element name="NumeroSerie">
        <h3>
            <meta:value dirty="true"/>
        </h3>
    </meta:element>
</meta:htmldescription>
```

C.3 Fichier *Meta* définitif du type de pages *Nouveautes*

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<meta:htmldescription xmlns:meta="http://www.cetic.be/FR/CRAQ-DB.htm">
    <meta:element name="Nouveautes">
        <html>
            <head>
                <meta:element ref="TitrePage"/>
                <meta/>
                <link/>
            </head>
            <body>
                <table>
                    <tr>
                        <td>
                            <meta:any/>
                            <!--colonne de gauche-->
                        </td>
                        <td>
                            <table>
                                    <meta:any/>
                                    <!--bandeau titre-->
                            </table>
                            <table>
                                    <meta:any/>
                                    <!--ligne Craq-Reverse-->
                            </table>
                            <meta:element ref="TitreContenu"/>
                            <table>
                                    <meta:element ref="Produit" max_card="N"/>
                            </table>
                        </td>
                    </tr>
                    <tr>
                        <meta:any/>
                        <!--fine ligne en-bas-->
                    </tr>
                </table>
            </body>
        </html>
    </meta:element>
    <meta:element name="TitrePage">
        <title>
            <meta:value/>
        </title>
    </meta:element>
    <meta:element name="TitreContenu">
        <h1>
            <meta:value/>
        </h1>
    </meta:element>
    <meta:element name="Produit">
        <tr>
            <td>
                <meta:element ref="NomProduit"/>
                <meta:element ref="Image"/>
                <meta:element ref="Personnalite"/>
                <meta:element ref="NombrePistes" min_card="0"/>
```

```
            <meta:element ref="Description"/>
            <h3>Pour en savoir plus...</h3>
            <p>
                <meta:any/>
            </p>
            <meta:element ref="NumeroSerie" min_card="0"/>
        </td>
    </tr>
</meta:element>
<meta:element name="NomProduit">
    <h2>
        <meta:value/>
    </h2>
</meta:element>
<meta:element name="Image">
    <meta:keyref>
        <a>
            <meta:value htmlAttribute="src">
                <img/>
            </meta:value>
        </a>
    </meta:keyref>
</meta:element>
<meta:element name="Personnalite">
    <meta:choice>
        <meta:group>
            <meta:element ref="Auteur"/>
        </meta:group>
        <meta:group>
            <meta:element ref="Realisateur"/>
        </meta:group>
        <meta:group>
            <meta:element ref="Interprete"/>
        </meta:group>
    </meta:choice>
</meta:element>-->
<meta:element name="Auteur">
    <h3>Auteur</h3>
    <p>
        <meta:value dirty="true"/>
    </p>
</meta:element>
<meta:element name="Interprete">
    <h3>Interprète</h3>
    <p>
        <meta:value dirty="true"/>
    </p>
</meta:element>
<meta:element name="Realisateur">
    <h3>Réalisateur</h3>
    <p>
        <meta:value dirty="true"/>
    </p>
</meta:element>
<meta:element name="Description">
    <h3>En bref...</h3>
    <p>
        <meta:value anyContent="true"/>
    </p>
```

```
        </meta:element>
        <meta:element name="NumeroSerie">
            <h3>
                <meta:value dirty="true"/>
            </h3>
        </meta:element>
        <meta:element name="NombrePistes">
            <h3>
                <meta:value dirty="true"/>
            </h3>
        </meta:element>
    </meta:htmldescription>
```

Annexe D

D.1 *XML Schema* du type de pages *Nouveautes* (version 1)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="NouveautesList">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="Nouveautes" maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="Nouveautes">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="TitrePage"/>
                <xsd:element ref="TitreContenu"/>
                <xsd:element ref="Produit" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="PageURL" type="xsd:string" use="required"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="TitrePage" type="xsd:string"/>
    <xsd:element name="TitreContenu" type="xsd:string"/>
    <xsd:element name="Produit">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="NomProduit"/>
                <xsd:element ref="Image"/>
                <xsd:element ref="Personnalite"/>
                <xsd:element ref="NombrePistes" minOccurs="0"/>
                <xsd:element ref="Description"/>
                <xsd:element ref="NumeroSerie" minOccurs="0"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="NomProduit" type="xsd:string"/>
    <xsd:element name="Image">
        <xsd:complexType>
            <xsd:simpleContent>
                <xsd:extension base="xsd:string">
                    <xsd:attribute name="keyref" type="xsd:string" use="required"/>
                </xsd:extension>
            </xsd:simpleContent>
        </xsd:complexType>
    </xsd:element>
```

```
    <xsd:element name="Personnalite">
        <xsd:complexType>
            <xsd:choice>
                <xsd:element ref="Auteur"/>
                <xsd:element ref="Realisateur"/>
                <xsd:element ref="Interprete"/>
            </xsd:choice>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="Auteur">
        <xsd:complexType>
            <xsd:simpleContent>
                <xsd:extension base="xsd:string">
                    <xsd:attribute name="dirty" type="xsd:boolean" use="required"/>
                </xsd:extension>
            </xsd:simpleContent>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="Interprete">
        <xsd:complexType>
            <xsd:simpleContent>
                <xsd:extension base="xsd:string">
                    <xsd:attribute name="dirty" type="xsd:boolean" use="required"/>
                </xsd:extension>
            </xsd:simpleContent>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="Realisateur">
        <xsd:complexType>
            <xsd:simpleContent>
                <xsd:extension base="xsd:string">
                    <xsd:attribute name="dirty" type="xsd:boolean" use="required"/>
                </xsd:extension>
            </xsd:simpleContent>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="Description">
        <xsd:complexType mixed="true">
            <xsd:sequence>
                <xsd:any processContents="skip" maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="NumeroSerie">
        <xsd:complexType>
            <xsd:simpleContent>
                <xsd:extension base="xsd:string">
                    <xsd:attribute name="dirty" type="xsd:boolean" use="required"/>
                </xsd:extension>
            </xsd:simpleContent>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="NombrePistes">
        <xsd:complexType>
            <xsd:simpleContent>
                <xsd:extension base="xsd:string">
                    <xsd:attribute name="dirty" type="xsd:boolean" use="required"/>
                </xsd:extension>
            </xsd:simpleContent>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

D.2 Données du type de pages *Nouveautes* (version 1)

```
<?xml version="1.0" encoding="UTF-8"?>
<NouveautesList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchema-
Location="Nouveautes.xsd">
    <Nouveautes PageURL="http://www.info.fundp.ac.be/~jrm/LittleZon/Nouveautes/NewLivres.htm">
        <TitrePage>LittleZon: Les nouveaux livres</TitrePage>
        <TitreContenu>LittleBookZon: les
nouveautés</TitreContenu>
        <Produit>
            <NomProduit>Harry Potter et l'Ordre du Phoenix</NomProduit>
            <Image keyref="../Livres/L0010.htm">../images/smallimgL0010.jpg</Image>
            <Personnalite>
                <Auteur dirty="true">Joanne-K. Rowling</Auteur>
            </Personnalite>
            <Description>Probablement une des sorties les plus attendues de l'histoire de
la littérature... voilà enfin en version française le 5ème volume
des aventures du plus célèbre des apprentis sorciers. Plus de 900
pages de plaisir et de suspense qui vous feront découvrir les
premiers pas de <i>Harry</i> dans l'adolescence. Ca valait la peine
d'attendre.</Description>
            <NumeroSerie dirty="true">Numéro dans la série : 5</NumeroSerie>
        </Produit>
        <Produit>
            <NomProduit>Antéchrista</NomProduit>
            <Image keyref="../Livres/L0011.htm">../images/smallimgL0011.jpg</Image>
            <Personnalite>
                <Auteur dirty="true">Amélie Nothomb</Auteur>
            </Personnalite>
            <Description>Un grand cri de souffrance et d'espoir pour les adolescents en
quête de reconnaissance et d'une identité propre.
 Un hymne à la liberté face à la tyranie et la dictature des
dominateurs quels qu'ils soient; la vie et rien d'autre la revanche
de l'opprimé sur l'oppresseur .</Description>
        </Produit>
    </Nouveautes>
    <Nouveautes PageURL="http://www.info.fundp.ac.be/~jrm/LittleZon/Nouveautes/NewDVDs.htm">
        <TitrePage>LittleZon: Les nouveautés DVD</TitrePage>
        <TitreContenu>LittleDVDZon: les
nouveautés</TitreContenu>
        <Produit>
            <NomProduit>Le Seigneur des Anneaux: Les Deux Tours</NomProduit>
            <Image keyref="../DVDs/D0011.htm">../images/smallimgD0011.jpg</Image>
            <Personnalite>
                <Realisateur dirty="true">Peter Jackson</Realisateur>
            </Personnalite>
            <Description>Dans l'attente frénétique de la sortie prévue le 17 décembre
2003 du troisième et dernier volet de la trilogie intitulé "Le
Retour du Roi", l'arrivée de ce titre va vous permettre de
(re)découvrir la version cinéma du second tableau de la fresque de
J.R.R Tolkien.
 Retrouvez Frodon et Sam et savourez leur curieuse rencontre avec
Gollum.</Description>
        </Produit>
        <Produit>
            <NomProduit>Mystic River</NomProduit>
            <Image keyref="../DVDs/D0009.htm">../images/smallimgD0009.jpg</Image>
            <Personnalite>
```

```
                <Realisateur dirty="true">Clint Eastwood</Realisateur>
            </Personnalite>
            <Description>Probablement le film le plus ambitieux et le plus réussi
d'Eastwood depuis longtemps, c'est également son film le plus noir
et le plus profondément désespéré à ce jour. Sur le canevas
faussement classique du film policier, Eastwood nous livre en fait
une peinture désenchantée de la nature humaine. On se laisse porter
par ces personnages désabusés jusqu'à un final d'une noirceur
absolue... et c'est tout simplement sublime.</Description>
        </Produit>
        <Produit>
            <NomProduit>Pirates des Caraïbes</NomProduit>
            <Image keyref="../DVDs/D0010.htm">../images/smallimgD0010.jpg</Image>
            <Personnalite>
                <Realisateur dirty="true">Gore Verbinski</Realisateur>
            </Personnalite>
            <Description>Inspiré d'une des attractions préférées des visiteurs des parcs
Walt Disney, ce film vous plongera dans une ambiance de flibuste
digne des grandes aventures maritimes des corsaires et autres
pirates.
 Un Johnny Depp au meilleur de sa forme et des décors sublimissimes
achèveront de vous séduire.</Description>
        </Produit>
    </Nouveautes>
    <Nouveautes PageURL="http://www.info.fundp.ac.be/~jrm/LittleZon/Nouveautes/NewCDs.htm">
        <TitrePage>LittleZon: Les nouveaux livres</TitrePage>
        <TitreContenu>LittleCDZon: les
nouveautés</TitreContenu>
        <Produit>
            <NomProduit>Quelqu'un m'a dit</NomProduit>
            <Image keyref="../CDs/C0009.htm">../images/smallimgC0009.jpg</Image>
            <Personnalite>
                <Interprete dirty="true">Carla Bruni</Interprete>
            </Personnalite>
            <NombrePistes dirty="true">Nombre de titres : 12</NombrePistes>
            <Description>Quand un top-model comme Carla Bruni s'essaie à la chanson,
toute la presse l'attend au tournant, afin de pouvoir la rouler
dans la farine, sortir les quolibets et autres blagues au goût
douteux. Cette fois, ils en ont pris pour leur grade, car la belle
plante ne se contente pas d'être parfaitement dessinée, elle a en
sus un fameux talent. Notre coup de coeur pour cette année.</Description>
        </Produit>
        <Produit>
            <NomProduit>Y.H.</NomProduit>
            <Image keyref="../CDs/C0010.htm">../images/smallimgC0010.jpg</Image>
            <Personnalite>
                <Interprete dirty="true">Yannick Noah</Interprete>
            </Personnalite>
            <NombrePistes dirty="true">Nombre de titres : 12</NombrePistes>
            <Description>Quatrième album déjà de l'ancien champion de tennis... Une seule
chose à dire, qu'il continue car à chaque opus la qualité et
l'intérêt de sa musique (et des paroles) ne font qu'augmenter...
jusqu'à ce qu'un jour, peut-être, sa victoire à Roland-Garros n'en
devienne anecdotique.</Description>
        </Produit>
    </Nouveautes>
</NouveautesList>
```

D.3 *XML Schema* du type de pages *Nouveautes* (final)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="NouveautesList">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="Nouveautes" maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="Nouveautes">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="TitrePage"/>
                <xsd:element ref="TitreContenu"/>
                <xsd:element ref="Produit" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="PageURL" type="xsd:string" use="required"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="TitrePage" type="xsd:string"/>
    <xsd:element name="TitreContenu" type="xsd:string"/>
    <xsd:element name="Produit">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="NomProduit"/>
                <xsd:element ref="Image"/>
                <xsd:element ref="Personnalite"/>
                <xsd:element ref="NombrePistes" minOccurs="0"/>
                <xsd:element ref="Description"/>
                <xsd:element ref="NumeroSerie" minOccurs="0"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="NomProduit" type="xsd:string"/>
    <xsd:element name="Image">
        <xsd:complexType>
            <xsd:simpleContent>
                <xsd:extension base="xsd:string">
                    <xsd:attribute name="keyref" type="xsd:string" use="required"/>
                </xsd:extension>
            </xsd:simpleContent>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="Personnalite">
        <xsd:complexType>
            <xsd:choice>
                <xsd:element ref="Auteur"/>
                <xsd:element ref="Realisateur"/>
                <xsd:element ref="Interprete"/>
            </xsd:choice>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="Description">
        <xsd:complexType mixed="true">
            <xsd:sequence>
                <xsd:any processContents="skip" maxOccurs="unbounded"/>
```

```
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="NumeroSerie" type="xsd:string"/>
    <xsd:element name="NombrePistes" type="xsd:string"/>
    <xsd:element name="AuteurChild_1" type="xsd:string"/>
    <xsd:element name="AuteurChild_2" type="xsd:string"/>
    <xsd:element name="Auteur">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="AuteurChild_1"/>
                <xsd:element ref="AuteurChild_2"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="RealisateurChild_1" type="xsd:string"/>
    <xsd:element name="RealisateurChild_2" type="xsd:string"/>
    <xsd:element name="Realisateur">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="RealisateurChild_1"/>
                <xsd:element ref="RealisateurChild_2"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="InterpreteChild_1" type="xsd:string"/>
    <xsd:element name="InterpreteChild_2" type="xsd:string"/>
    <xsd:element name="Interprete">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="InterpreteChild_1"/>
                <xsd:element ref="InterpreteChild_2"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

D.4 Données nettoyées du type de pages *Nouveautes*


```
<?xml version="1.0" encoding="UTF-8"?>
<NouveautesList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchema-
Location="Nouveautes.xsd">
    <Nouveautes PageURL="http://www.info.fundp.ac.be/~jrm/LittleZon/Nouveautes/NewLivres.htm">
        <TitrePage>LittleZon: Les nouveaux livres</TitrePage>
        <TitreContenu>LittleBookZon: les
nouveautés</TitreContenu>
        <Produit>
            <NomProduit>Harry Potter et l'Ordre du Phoenix</NomProduit>
            <Image keyref="../Livres/L0010.htm">../images/smallimgL0010.jpg</Image>
            <Personnalite>
                <Auteur>
                    <AuteurChild_1>Joanne-K.</AuteurChild_1>
                    <AuteurChild_2>Rowling</AuteurChild_2>
                </Auteur>
            </Personnalite>
            <Description>Probablement une des sorties les plus attendues de l'histoire de
la littérature... voilà enfin en version française le 5ème volume
des aventures du plus célèbre des apprentis sorciers. Plus de 900
pages de plaisir et de suspense qui vous feront découvrir les
premiers pas de <i>Harry</i> dans l'adolescence. Ca valait la peine
d'attendre.</Description>
            <NumeroSerie>5</NumeroSerie>
        </Produit>
        <Produit>
            <NomProduit>Antéchrista</NomProduit>
            <Image keyref="../Livres/L0011.htm">../images/smallimgL0011.jpg</Image>
            <Personnalite>
                <Auteur>
                    <AuteurChild_1>Amélie</AuteurChild_1>
                    <AuteurChild_2>Nothomb</AuteurChild_2>
                </Auteur>
            </Personnalite>
            <Description>Un grand cri de souffrance et d'espoir pour les adolescents en
quête de reconnaissance et d'une identité propre.
 Un hymne à la liberté face à la tyranie et la dictature des
dominateurs quels qu'ils soient; la vie et rien d'autre la revanche
de l'opprimé sur l'oppresseur .</Description>
        </Produit>
    </Nouveautes>
    <Nouveautes PageURL="http://www.info.fundp.ac.be/~jrm/LittleZon/Nouveautes/NewDVDs.htm">
        <TitrePage>LittleZon: Les nouveautés DVD</TitrePage>
        <TitreContenu>LittleDVDZon: les
nouveauté</TitreContenu>
        <Produit>
            <NomProduit>Le Seigneur des Anneaux: Les Deux Tours</NomProduit>
            <Image keyref="../DVDs/D0011.htm">../images/smallimgD0011.jpg</Image>
            <Personnalite>
                <Realisateur>
                    <RealisateurChild_1>Peter</RealisateurChild_1>
                    <RealisateurChild_2>Jackson</RealisateurChild_2>
                </Realisateur>
            </Personnalite>
            <Description>Dans l'attente frénétique de la sortie prévue le 17 décembre
2003 du troisième et dernier volet de la trilogie intitulé "Le
Retour du Roi", l'arrivée de ce titre va vous permettre de
```

```
(re)découvrir la version cinéma du second tableau de la fresque de
J.R.R Tolkien.
 Retrouvez Frodon et Sam et savourez leur curieuse rencontre avec
Gollum.</Description>
        </Produit>
        <Produit>
            <NomProduit>Mystic River</NomProduit>
            <Image keyref="../DVDs/D0009.htm">../images/smallimgD0009.jpg</Image>
            <Personnalite>
                <Realisateur>
                    <RealisateurChild_1>Clint</RealisateurChild_1>
                    <RealisateurChild_2>Eastwood</RealisateurChild_2>
                </Realisateur>
            </Personnalite>
            <Description>Probablement le film le plus ambitieux et le plus réussi
d'Eastwood depuis longtemps, c'est également son film le plus noir
et le plus profondément désespéré à ce jour. Sur le canevas
faussement classique du film policier, Eastwood nous livre en fait
une peinture désenchantée de la nature humaine. On se laisse porter
par ces personnages désabusés jusqu'à un final d'une noirceur
absolue... et c'est tout simplement sublime.</Description>
        </Produit>
        <Produit>
            <NomProduit>Pirates des Caraïbes</NomProduit>
            <Image keyref="../DVDs/D0010.htm">../images/smallimgD0010.jpg</Image>
            <Personnalite>
                <Realisateur>
                    <RealisateurChild_1>Gore</RealisateurChild_1>
                    <RealisateurChild_2>Verbinski</RealisateurChild_2>
                </Realisateur>
            </Personnalite>
            <Description>Inspiré d'une des attractions préférées des visiteurs des parcs
Walt Disney, ce film vous plongera dans une ambiance de flibuste
digne des grandes aventures maritimes des corsaires et autres
pirates.
 Un Johnny Depp au meilleur de sa forme et des décors sublimissimes
achèveront de vous séduire.</Description>
        </Produit>
    </Nouveautes>
    <Nouveautes PageURL="http://www.info.fundp.ac.be/~jrm/LittleZon/Nouveautes/NewCDs.htm">
        <TitrePage>LittleZon: Les nouveaux livres</TitrePage>
        <TitreContenu>LittleCDZon: les
nouveautés</TitreContenu>
        <Produit>
            <NomProduit>Quelqu'un m'a dit</NomProduit>
            <Image keyref="../CDs/C0009.htm">../images/smallimgC0009.jpg</Image>
            <Personnalite>
                <Interprete>
                    <InterpreteChild_1>Carla</InterpreteChild_1>
                    <InterpreteChild_2>Bruni</InterpreteChild_2>
                </Interprete>
            </Personnalite>
            <NombrePistes>12</NombrePistes>
            <Description>Quand un top-model comme Carla Bruni s'essaie à la chanson,
toute la presse l'attend au tournant, afin de pouvoir la rouler
dans la farine, sortir les quolibets et autres blagues au goût
douteux. Cette fois, ils en ont pris pour leur grade, car la belle
plante ne se contente pas d'être parfaitement dessinée, elle a en
sus un fameux talent. Notre coup de coeur pour cette année.</Description>
```

```
        </Produit>
        <Produit>
            <NomProduit>Y.H.</NomProduit>
            <Image keyref="../CDs/C0010.htm">../images/smallimgC0010.jpg</Image>
            <Personnalite>
                <Interprete>
                    <InterpreteChild_1>Yannick</InterpreteChild_1>
                    <InterpreteChild_2>Noah</InterpreteChild_2>
                </Interprete>
            </Personnalite>
            <NombrePistes>12</NombrePistes>
            <Description>Quatrième album déjà de l'ancien champion de tennis... Une seule
chose à dire, qu'il continue car à chaque opus la qualité et
l'intérêt de sa musique (et des paroles) ne font qu'augmenter...
jusqu'à ce qu'un jour, peut-être, sa victoire à Roland-Garros n'en
devienne anecdotique.</Description>
        </Produit>
        <Produit>
            <NomProduit>Boucan d'enfer</NomProduit>
            <Image keyref="../CDs/C0008.htm">../images/smallimgC0008.jpg</Image>
            <Personnalite>
                <Interprete>
                    <InterpreteChild_1>Renaud</InterpreteChild_1>
                    <InterpreteChild_2>Sechan</InterpreteChild_2>
                </Interprete>
            </Personnalite>
            <NombrePistes>14</NombrePistes>
            <Description>On n'y croyait plus. Renaud avait disparu de la circulation
depuis un bon moment et il revient en fanfare avec ce superbe album
qui raconte notamment son parcours difficile de ces dernières
années. On en retiendra plus des textes puissants et une musique
dans la lignée de ses autres albums, plutôt que la qualité de sa
voix, qui, si elle émeut à souhait, ne brille pas par sa justesse
ni sa clarté.</Description>
        </Produit>
    </Nouveautes>
</NouveautesList>
```

Annexe E

E.1Code DDL de création de la base de données

```
-- ********************************************
-- * Standard SQL generation                  *
-- *------------------------------------------*
-- * Generator date:  Dec 12 2003             *
-- * Generation date: Mon May 24 11:28:54 2004 *
-- ********************************************

-- Database Section
-- ________________

create database LittleZon;

-- DBSpace Section
-- _______________

-- Tables Section
-- ______________

create table anime (
    Reference char(5) not null,
    PersID numeric(5) not null,
    primary key (Reference, PersID));

create table CD (
    Reference char(5) not null,
    PersID numeric(5) not null,
    primary key (Reference));

create table DVD (
    Reference char(5) not null,
    Synopsis varchar(255) not null,
    Duree varchar(6) not null,
    primary key (Reference));

create table joue (
    Reference char(5) not null,
    PersID numeric(5) not null,
    primary key (Reference, PersID));

create table Livre (
    Reference char(5) not null,
    Resume varchar(255) not null,
    Editeur varchar(30) not null,
    Extrait varchar(255),
    NumeroSerie numeric(2),
    PersID numeric(5) not null,
```

```
    primary key (Reference));

create table NouveauProduit (
    Reference char(5) not null,
    Description varchar(255) not null,
    primary key (Reference));

create table Personnalite (
    PersID numeric(5) not null,
    Prenom varchar(20) not null,
    Nom varchar(20) not null,
    primary key (PersID));

create table Piste (
    Reference char(5) not null,
    Numero numeric(5) not null,
    Piste varchar(50) not null,
    primary key (Numero, Reference));

create table Produit (
    Reference char(5) not null,
    NomProduit varchar(50) not null,
    Image varchar(100) not null,
    Annee numeric(4),
    Genre varchar(20),
    Avis varchar(6) not null,
    Livre char(5),
    DVD char(5),
    CD char(5),
    primary key (Reference));

create table Reaction (
    ReacID numeric(5) not null,
    Identite varchar(30) not null,
    Critique varchar(255) not null,
    Reference char(5) not null,
    primary key (ReacID));

create table realise (
    Reference char(5) not null,
    PersID numeric(5) not null,
    primary key (Reference, PersID));

-- Constraints Section
-- ___________________

alter table anime add constraint FKani_DVD
    foreign key (Reference)
    references DVD;

alter table anime add constraint FKani_Per
    foreign key (PersID)
    references Personnalite;

--alter table CD add constraint
--    check(exists(select * from Piste
--                 where Piste.Reference = Reference));

alter table CD add constraint FKPro_CD
    foreign key (Reference)
    references Produit;

alter table CD add constraint FKchante
    foreign key (PersID)
    references Personnalite;
```

```
--alter table DVD add constraint
--    check(exists(select * from realise
--                 where realise.Reference = Reference));

alter table DVD add constraint FKPro_DVD
    foreign key (Reference)
    references Produit;

alter table joue add constraint FKjou_DVD
    foreign key (Reference)
    references DVD;

alter table joue add constraint FKjou_Per
    foreign key (PersID)
    references Personnalite;

alter table Livre add constraint FKecrit
    foreign key (PersID)
    references Personnalite;

alter table Livre add constraint FKPro_Liv
    foreign key (Reference)
    references Produit;

alter table NouveauProduit add constraint FKkeyref
    foreign key (Reference)
    references Produit;

alter table Piste add constraint FKcompose
    foreign key (Reference)
    references CD;

--alter table Produit add constraint EXCL_Produit
--    check((CD is not null and DVD is null and Livre is null)
--          or (CD is null and DVD is not null and Livre is null)
--          or (CD is null and DVD is null and Livre is not null)
--          or (CD is null and DVD is null and Livre is null));

alter table Reaction add constraint FKcritique
    foreign key (Reference)
    references Produit;

alter table realise add constraint FKrea_DVD
    foreign key (Reference)
    references DVD;

alter table realise add constraint FKrea_Per
    foreign key (PersID)
    references Personnalite;


-- Index Section
-- _____________

create unique index ID_anime
    on anime (Reference, PersID);

create index FKani_DVD
    on anime (Reference);

create index FKani_Per
    on anime (PersID);

create unique index FKPro_CD
    on CD (Reference);
```

```
create index FKchante
    on CD (PersID);

create unique index FKPro_DVD
    on DVD (Reference);

create unique index ID_joue
    on joue (Reference, PersID);

create index FKjou_DVD
    on joue (Reference);

create index FKjou_Per
    on joue (PersID);

create index FKecrit
    on Livre (PersID);

create unique index FKPro_Liv
    on Livre (Reference);

create unique index FKkeyref
    on NouveauProduit (Reference);

create unique index ID_Personnalite
    on Personnalite (PersID);

create unique index ID_Piste
    on Piste (Numero, Reference);

create index FKcompose
    on Piste (Reference);

create unique index ID_Produit
    on Produit (Reference);

create index GRProduit
    on Produit (NomProduit);

create index GRProduit_1
    on Produit (Annee);

create index GRProduit_2
    on Produit (Genre);

create unique index ID_Reaction
    on Reaction (ReacID);

create index FKcritique
    on Reaction (Reference);

create unique index ID_realise
    on realise (Reference, PersID);

create index FKrea_DVD
    on realise (Reference);

create index FKrea_Per
    on realise (PersID);
```